CODEN: LUTFD2/(TFRT-5400)/1-85/(1989)

Styrning av frekvensanalysator

Eric Wichtel

Institutionen för Reglerteknik
Lunds Tekniska Högskola
April 1989

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Master Thesis

Date of issue
April 1989

Document Number
CODEN: LUTFD2/(TFRT-5400)/1-85/(1989)

Author(s)
Eric Wichtel

Supervisor
Rolf Johansson

Sponsoring organisation

Title and subtitle
Styrning av frekvensanalysator (Control of frequency analyzer.)

Abstract

The method to obtain mathematical models from experimental data is called 'process identification'. This has great significance in the area of automatic control. Often, control design is based upon the mathematical transfer function, which describes the process to be controlled.

Frequency response analysis is one of the most important tools of identification. This analysis is performed by sending a range of different sinus signals as input to the process. Then the quotient between the output signal and the input signal is studied. For each input frequency we obtain a specific value of the transfer function. A frequency response analyser is used with such experiments. When in use the analyser, available at the Department of Automatic Control, (LTH), needs to be controlled by a computer. This Master Thesis consists of such a control program that is easy to use. It collects measurement data, presents data in Bode plots and prepares for further data processing in the package 'PC-MATLAB'. The program is intended for research purposes as well as for education.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
85

Recipient's notes

Security classification

0. Inledning

0.1

Processidentifiering innebär att man ur experimentdata skattar matematiska modeller. Detta har stor betydelse inom reglertekniken. Man konstruerar ofta sina regulatorer utifrån överföringsfunktioner, som beskriver den reglerade processen. Att finna överföringsfunktionen är här identifieringsuppgiften. Frekvensanalys är ett av identifieringens viktigaste redskap. Analysen utförs genom att skicka insignaler med olika frekvenser till processen och sedan studera utsignalen dividerad med insignalen. Varje sådan mätpunkt blir den sökta överföringsfunktionens värde vid aktuell mätfrekvens. Som hjälpmedel härför har man en frekvensanalysator. Den frekvensanalysator (Solartron 1250) som finns vid Institutionen för Reglerteknik i Lund behöver dock styras av en dator för att kunna användas praktiskt. Detta examensarbete består av ett program, som på ett enkelt och användarvänligt sätt styr Solartronens mest använda funktioner, tar hand om mätdata, presenterar data i Bodediagram samt gör det möjligt att använda mätdata i PC-Matlab. Programmet är avsett att användas för undervisning i kursen Processidentifiering samt för forskning.

0.2 INNEHÅLLSFÖRTECKNING

1. PROBLEMSTÄLLNING

1.1 BEFINTLIGA SYSTEM

Det finns ett existerande styrsystem för frekvensanalysatorn (förkortas framöver FRA), i form av ett basic-program och en AppleII-dator. Systemet är omodernt, användarovänligt och mätdata kan inte vidarebehandlas med befintlig mjukvara. Programmet är omfattande, vilket för enklare användningar gör bruket onödigt komplicerat.

Andra alternativ finns också, såsom "IBM PERSONAL COMPUTER INTERFACING VIA HEWLETT-PACKARD HPIB". Programmet fås från tillverkaren av analysatorn och är även det skrivet i basic. Denna lösning uppfyller dock inte funktionskraven enligt punkt 1.2

1.2 SPECIFIKATION AV NYTT SYSTEM

Institutionen vill ha ett system som integrerar mätning- datainsamling-analys. Det skall vara användarvänligt och ha fördefinierade parametrar, som överensstämmer med institutionens utrustningar i möjligaste mån. Det önskade systemet skall inte bara utföra frekvensanalys-experimentet utan även innehålla designverktyg. Se 2.4. Programmet skall skrivas i MODULA 2 för IBM, eftersom språket allmänt används på institutionen och åtskilliga programbibliotek finns.

Funktionskrav:

- Kommunicera med SOLARTRON 1250 frekvensanalysator via serielänk.
- Parametrar för frekvensanalysexperimentet skall kunna sättas via olika menyer. Menyerna väljs med musen.
- En parameteruppsättning för frekvensanalysatorn skall kunna lagras på editerbar fil. Filen skall även kunna läsas av programmet, så att t.ex. en uppsättning default-parametrar kan användas.
- Efter ett experiment skall mätresultatets Bodediagram eller Nyquistdiagram kunna ritas på skärmen.
- Efter ett experiment skall ett nytt experiment kunna göras och data från detta skall om så önskas adderas till tidigare mätningar.
- Uppmätta data skall kunna skrivas ut på editerbar fil.
- Alla filer skall vara editerbara och i förekommande fall vara läsbara av PC-MATLAB. Filnamn skall kunna väljas av användaren.

2. LÖSNING

2.1 RESULTATET

Ett helt nytt program enligt specifikationen (1.2) har utvecklats. En hastig genomläsning av detta kapitel ger nog intrycket att här står samma sak som i specifikationen. Flera skillnader gentemot specifikationen finns dock. Koden är skriven i Logitechs Modula-2 på en IBM-AT med EGA-grafik och mus. Institutionens programbibliotek för grafik, kommunikation via seriell port samt realtidskärna har använts.

- Resultatet har blivit ett meny(mus-)styrt program, där de av FRA:ns kommandon som är intressanta ur användarsynpunkt kan sättas. Styrningsmöjligheterna är alltså till en viss grad begränsade.
- FRA:ns parameteruppsättning kan sparas eller läsas in från editerbar fil. Skulle någon parameter behöva sättas, som inte finns i menyerna, kan kommandot editeras in i en parameterfil och sedan läsas in av programmet.
- Ett analysexperiment i form av ett svep (eller flera efter varandra, med olika parameteruppsättningar) kan utföras. s då in i tur och ordning
- Filnamnen på parameterfilerna kan i sin tur lagras på en och sorteras in i samma resultatfil. och mätning utförs, så att flera mätningar sker efter hand och sorteras in i samma resultatfil.
- Mätresultatet lagras alltid på editerbar fil. Mätdata kan därför inte gå förlorade även om oförutsedda saker inträffar. Denna resultatfil kan sparas under eget filnamn samt läsas in på nytt.
- Ny mätning som sorteras samman med en gammal resultatfil kan utföras.
- Man kan få Bodediagram på sin resultatfil. Resultatfilen kan läsas av PC-MATLAB, vilket ger vidare möjligheter till mätdatabehandling såsom Bode- och Nyquistdiagram. Diagramfunktionen har inte gjorts så avancerad, eftersom den redan finns i PC-MATLAB. Den behöver bara användas t.ex. som kontroll att mätningen inte varit för gles. Avancerade analyser gör man i Matlab.
- Ordentlig hjälpfunktion finns inbyggd i programmet.

I övrigt hänvisar jag till manualen och programdokumentationen.

2.2 UTVECKLINGSMÖJLIGHETER

Detta avsnitt bör läsas först när man känner systemet.

2.2.1

En rad ytterligare finesser i programmet vore tänkbara:

- CHAIN utökas med möjlighet att få delmätningarnas resultat på olika filer.
- Editeringsmöjligheter i CHAIN så att man slipper göra små ändringar i en editor utanför programmet.
- Ett DOS SHELL skulle kunna ge möjlighet att editera filer utan att lämna programmet.
- Möjligheter att ge ytterligare kommandon till FRA:n. Man kan också tänka sig någon slags on-line kommunikation med FRA:n, men då får man problem med att veta vad parametrarna egentligen är satta till.

Programmet skulle också kunna innehålla designverktyg och provningsmöjligheter för den erhållna regulatorn.

Följande utvecklingsmöjligheter finns enligt specifikationen av examensarbetet:

2.2.2 DESIGN AV KOMPENSERINGSLÄNKAR

- Det erhållna bodediagrammet skall kunna användas för design av kompenseringslänkar. Sådana skall kunna adderas, och även bodediagrammet för den erhållna kretsöverföringen skall kunna ritas ut. Även det öppna systemets bodediagram skall visas om så önskas. Viktiga hjälplinjer skall ritas ut i bodediagrammet.
- Kretsen skall kunna slutas. Förfilter skall kunna bestämmas för att ge önskade egenskaper åt överföringsfunktionen.
- Erhållen kompenseringslänk och förfilter skall kunna skrivas ut på editerbar fil.

2.2.3 REGULATOR

- Efter det att förfilter och kompenseringslänk bestämts, skall en regulatoralgoritm implementera dessa.
- Regulatorns parametrar skall kunna lagras på fil för att sedan åter kunna läsas in till regulatorn.
- En referensvärdesgenerator skall kunna ge en fyrkantvåg.
- Plottning skall kunna ske av referensvärde, insignal och utsignal till systemet.

2.3 EN SESSION MED SYSTEMET

Vi skall här titta lite närmare på hur ett enkelt frekvensanalysexperiment går till med det nya systemet. Processen vi ska mäta på är en av institutionens laboratorie- processer, nämligen "bommen". Denna består av just en bom som kan vridas med en motor. Bommens vinkelläge kan mätas. Vi undrar därför som ett led i identifieringen av processen, vilket ordningstal systemet har.

- Förberedelser

Datorn och frekvensanalysatorn är ihopkopplade via en seriell kabel. Processen har som insignal den utsignal, som FRA:ns generatordel lämnar. Generatorn är även sammankopplad med analysatorns kanal 1. Analysatorns kanal 2 mäter processens utsignal. Internt dividerar sedan FRA:n kanal 2 med kanal 1 och som mätresultat kommer vi då att få värden på överföringsfunktionen vid olika frekvenser.

- Uppstart

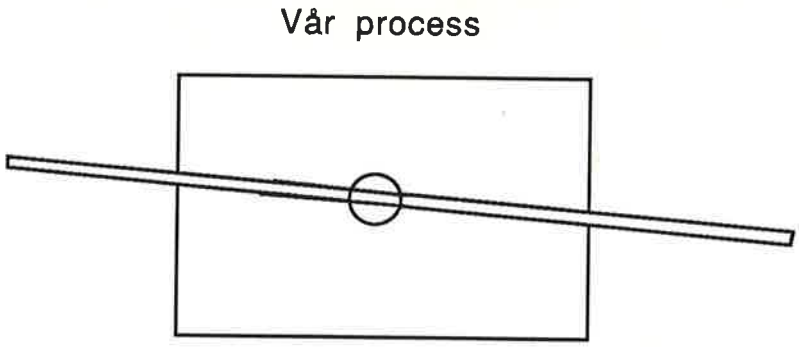
Slå på strömmen till process, frekvensanalysator och dator. Med kommandot

```
C:\> PCFRA
```

startas styrprogrammet upp. FRA:n initialiseras automatiskt med en uppsättning standardparametrar, vilket förhoppningsvis leder till att endast ett minimum av manuella inställningar behöver göras.

- Inställningar

Vi befinner oss nu i huvudmenyn. Skärmutseendena finns samlade i Appendix 4. Där finns också några små användningsexempel. För att ställa in FRA:n klickar vi med musen i rutan MANUAL INITIALISATION och kommer till en ny meny (som heter just "manual initialisation").



MAIN MENU

INITIALISE: [klickl] MANUAL INITIALISATION

[ ] CHAIN IS OFF

[ ] SAVE/LOAD PARAMETERS

[ ] STATUS

MEASURE: [ ] NEW MEASUREMENT

[ ] ADD DATA FILES

RESULT- [ ] SAVE/LOAD MEASUREMENT DATA

OUTPUT: [ ] BODE DIAGRAM

[ ] HELP

[ ] BACK TO DOS

MANUAL INITIALISATION

GENERATOR:	☐	AMPLITUDE	0.5 V
	☐	BIAS	0.0 V
	☐	WAVEFORM	SINUS
SWEEP:	☐	MIN FREQUENCY	0.05 Hz
	☐	MAX FREQUENCY	50 Hz
	☐	LOG INCREMENT	10 STEPS/DECADE
ANALYSER:	☐	INTEGRATION TIME	5 CYCLES
	☐	MEASUREMENT DELAY	5 s

☐ HELP

☐ BACK TO MAIN MENU

Vi vill göra ett frekvenssvep mellan 0.05 Hz och 50 Hz och väljer därför MIN FREQUENCY respektive MAX FREQUENCY, samt matar in de rätta värdena. Analysatorn behöver MEASUREMENT DELAY på 5 sekunder för att processen ska hinna svänga in sig vid varje ny frekvens. Mot mätbrus väljer vi INTEGRATION till 5 cykler.

MANUAL INITIALISATION

GENERATOR:	☐	AMPLITUDE	0.5 V
	☐	BIAS	0.0 V
	☐	WAVEFORM	SINUS
SWEEP:	☐	MIN FREQUENCY	0.05 Hz
	☐	MAX FREQUENCY	50 Hz
	☐	LOG INCREMENT	10 STEPS/DECADE
ANALYSER:	■	INTEGRATION TIME	5 CYCLES
	☐	MEASUREMENT DELAY	5 s
	☐	HELP	
	☐	BACK TO MAIN MENU	

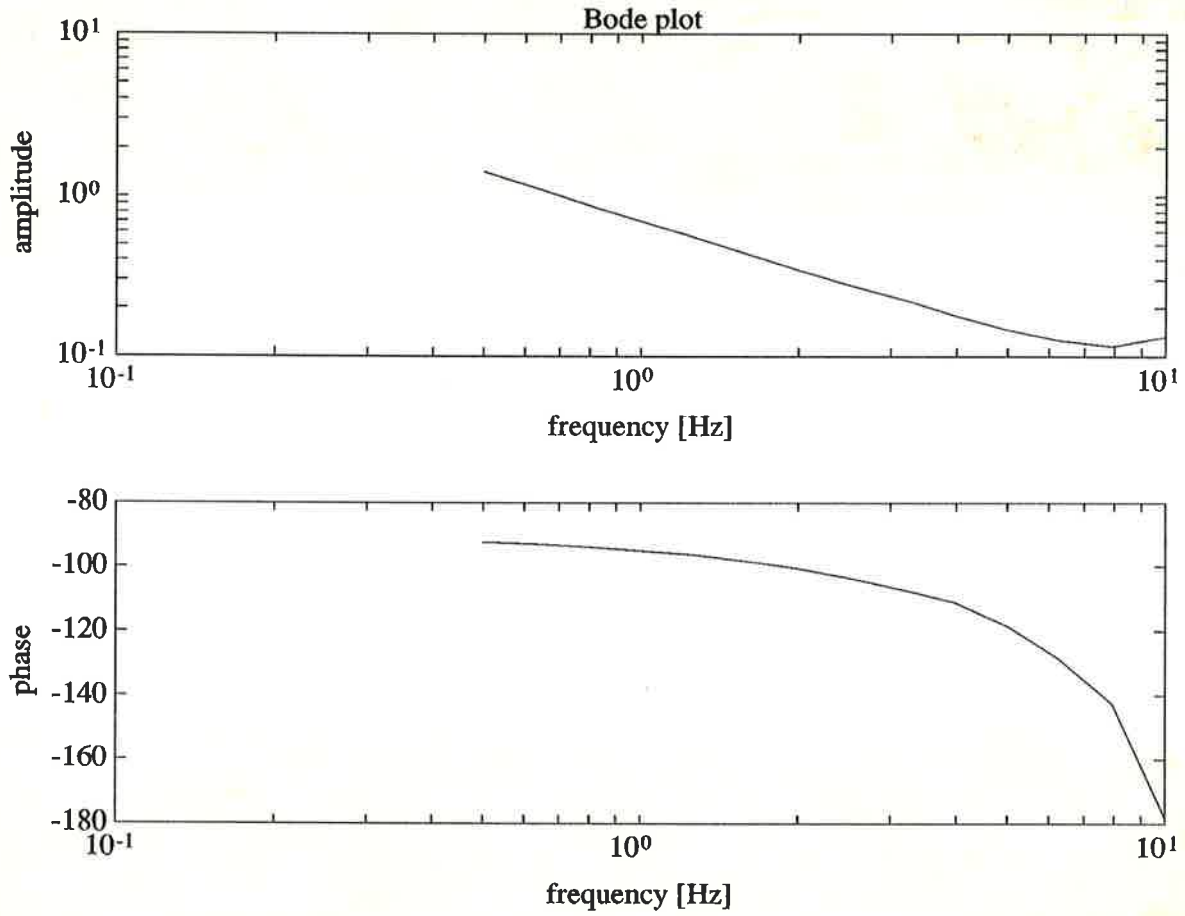
SETTING OF INTEGRATION
0=SECONDS
1=CYCLES
2=AUTO

NEW VALUE:

MANUAL INITIALISATION

GENERATOR:	☐	AMPLITUDE	0.5 V
	☐	BIAS	0.0 V
	☐	WAVEFORM	SINUS
SWEEP:	☐	MIN FREQUENCY	0.05 Hz
	☐	MAX FREQUENCY	50 Hz
	☐	LOG INCREMENT	10 STEPS/DECADE
ANALYSER:	■	INTEGRATION TIME	5 CYCLES
	☐	MEASUREMENT DELAY	5 s
	☐	HELP	
	☐	BACK TO MAIN MENU	

INTEGRATION TIME: VALUE IN CYCLES
NEW VALUE:



Figur 2.1 Bodediagram erhållet i PC-MATLAB

Övriga inställningar som amplitud (0.5 V), kanalnummer m.m. stämmer överens med de fördefinierade värdena och vi återgår därför till huvudmenyn genom klick i rutan "back to main menu". Hade vi gjort fler ändringar i inställningarna hade det kanske varit lönt att spara dessa på skiva, men nu går vi direkt vidare.

- Mätning

Klickar i rutan NEW MEASUREMENT. Mätningen utförs och någon minut senare är den klar. FRA:n piper till för att tala om att den är färdig. Vi kommer automatiskt tillbaka till huvudmenyn.

- Analys av resultatet

Bodediagram kan vi titta på direkt, klicka bara i rutan BODE DIAGRAM. Klicka en gång till för att komma tillbaka till menyn. Mätdata ligger i filen TEMP.MEA, om vi inte väljer att spara den under eget filnamn vid rubriken SAVE/LOAD MEASUREMENT DATA. PC-MATLAB heter ett program med mycket bättre analysmöjligheter, varför vi lämnar PC-FRA och bearbetar materialet där. Skriv

```
C:\>MATLAB
```

Vi kommer nu in i PC-MATLAB. Först skall mätdatafilen TEMP.MEA läsas in i matlab.

```
>LOAD TEMP.MEA
```

Matlab verifierar inläsningen, när den skett. Bodediagrammet fås sedan med kommandot

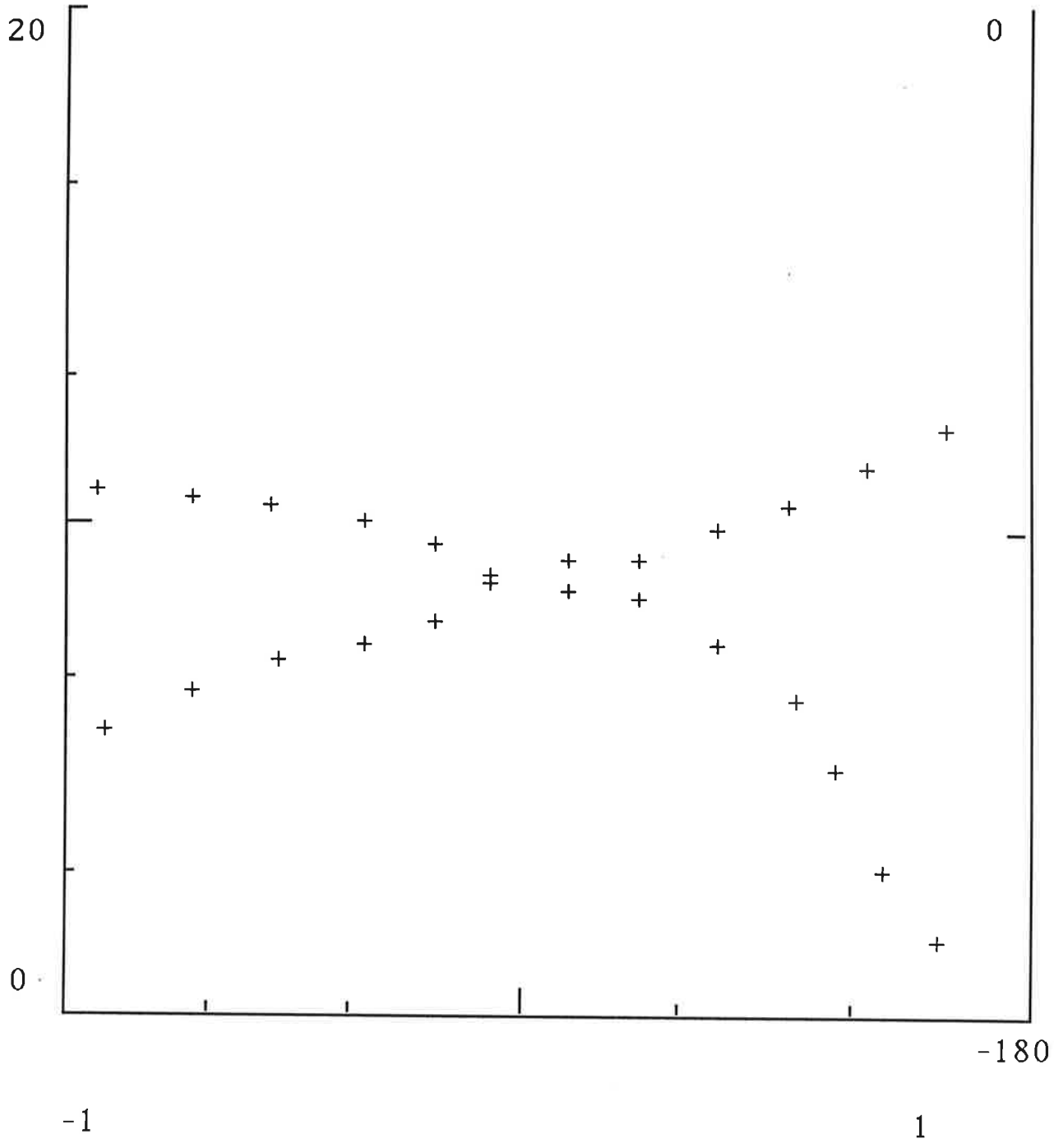
```
>BODEPLOT(TEMP)
```

Diagrammet består av separata amplitud och faskurvor, snygga och överskådliga. Tryck "return" och skriv sedan

```
>EXIT
```

för att lämna PC-MATLAB.

Principiellt utseende för resultatet av mätningen.



3. SLUTSATS

Det resulterande programmet uppfyller specifikationen och fungerar väl. Det förenklar hanteringen av FRA:n genom att endast nödvändiga inställningsmöjligheter presenteras för användaren. Enbart 12 av tidigare 98 parametrar finns kvar att ställa in. Dessutom är inställningen av dessa mera användarvänliga nu än tidigare. PC-FRA levererar mätdata till PC-MATLAB för analys. Jag tror att programmet kan användas ganska lätt även av dem som inte sysslat med frekvensanalys tidigare. Samtidigt hoppas jag att jag inte förenklat allting så långt, att det inte går att använda systemet för avancerade mätningar.

4. EFTERKLOKHETER

Vad kunde jag gjort bättre och vad har jag misslyckats med?

Programmet har blivit stort och borde från början ha indelats i moduler. Men inget program är stort och oöverskådligt, när det börjar skrivas. Och det visade sig senare inte möjligt att modularisera upp det med måttliga arbetsinsatser. Det är nog så att den pascalundervisning som ges på LTH dels inte ger erfarenhet av lite större program och dels inte uppmuntrar till att tänka i moduler. Den fällan faller man alltså lätt i. Flera av programavsnitten har jag i stället utvecklat i separata småprogram, som när de fungerat har kopierats in i det stora programmet. Denna variant är nästan lika bra för mig, men lämnar en del övrigt att önska för en eventuell efterföljare.

Ett par gånger har jag fått "procedure too large - implementation limit reached". Detta har tvingat mig att dela procedurerna utöver vad jag anser vara riktiga uppdelningar av koden. Antalet procedurer är ändå stort. (JA, det hade varit bra med fler moduler...)

"... eine Seele. Was ist das? - Es ist negativ leicht bestimmt: es ist eben das, was sich verkriecht, wenn man von algebraischen Reihen hört."
ROBERT MUSIL

Byt ut numeriska serier mot filer, så får Du reda på vad som ställt till bekymmer under programutvecklingen. Detta beror på att modula-2 i vissa fall omvandlar kontrolltecken, när man läser och skriver dem. Detta, tillsammans med filers inbyggda förmåga att helt enkelt bara krångla gav upphov till de flesta problem jag haft.

Jag har också funderat på en möjlighet av ha en slags direkt kommunikation med FRA:n, så att en kvalificerad användare kan ställa alla parametrar inifrån programmet. Någon enkel lösning till detta som är bra har jag inte hittat. Problemet är i stället löst genom att man med en editor kan skriva in kommandona i en parameterfil, som sedan läses av programmet.

Operatörskommunikation är ett område, som man kan lägga hur mycket tid som helst på. Programmet är säkert för normal användning, om man inte inuti PC-FRA försöker läsa filer som inte existerar.

Gör man BODE DIAGRAM och TEMP.MEA inte existerar kastas man ur programmet. T.ex. tar SAVE MEASUREMENT DATA bort TEMP.MEA.

Om man vid LOAD MEASUREMENT DATA väljer en fil som inte finns, protesterar inte programmet. Också här kastas man sedan ur om man försöker göra ett bodediagram. Annars händer ingenting.

SAVE MEASUREMENT DATA fungerar så att TEMP.MEA byter namn till det filnamn man valt. Om inte TEMP.MEA existerar, skapas en tom fil.

LOAD PARAMETERS på ett inte existerande filnamn får till följd att programmets information om FRA:ns inställningar går förlorade. FRA:n behåller ändå sin inställing. För att man skall kunna genomföra en mätning måste man sätta parametern MAX FREQUENCY eller ladda in en ny parameteruppsättning med LOAD PARAMETERS. Annars kastas man ut, när man trycker NEW MEASUREMENT.

Gör man LOAD CHAIN på ej existerande fil märker inte användaren något även i detta fall. Ett försök till CHAIN- mätning nu leder bara till att ingen mätning utförs.

APPENDIX

-Appendix 1
APPARATDOKUMENT

1. HÅRDVARULISTA

Del, Leverantör

1250 Frequency Responce Analyser, Solartron Instruments
IBM Personal Computer, IBM
IBM-PC EGA color monitor, IBM
IBM-PC enchanced graphics adaptor, IBM
RS423 seriell port+kabel, IBM m.fl.
Microsoft Mouse, Microsoft

2. MJUKVARULISTA

Del, Leverantör

PC-FRA, Reglerteknik, LTH

PC-DOS

Mouse Device Driver, Microsoft

3. KOPPLINGAR

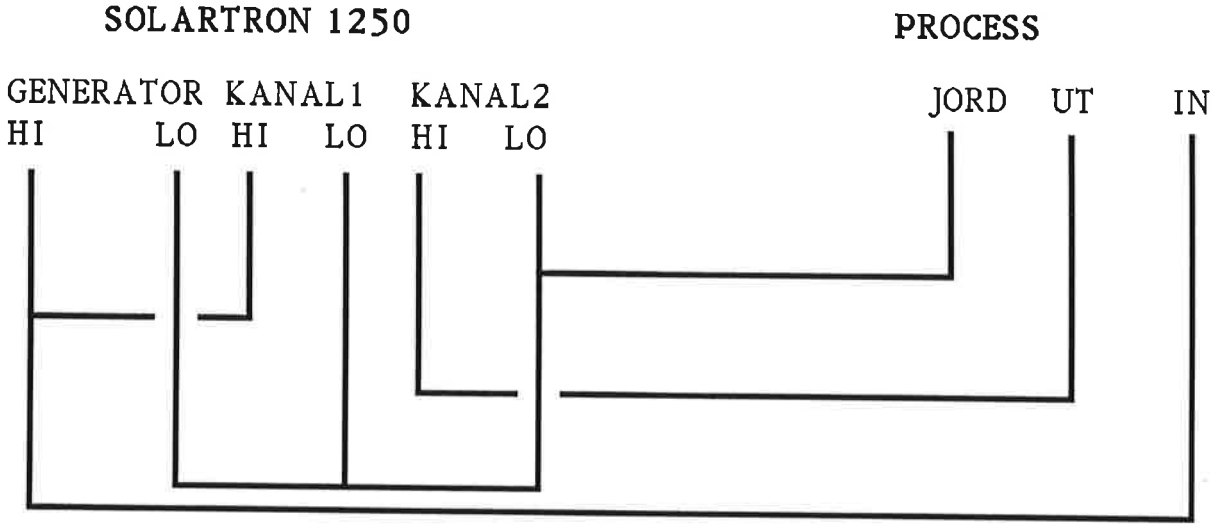
Analysatorn sammankopplas med IBM AT:n via den seriella porten.

Solartronens

- generatorutgång kopplas som processens insignal.
- kanal 1 kopplas till generatorn.
- kanal 2 kopplas in på processens utsignal.

Glöm inte jorda.

EXPERIMENTUPPKOPPLING



-Appendix 2 PCFRA.MOD, PROGRAMDOKUMENTATION

Programmet är skrivet i Logitech Modula-2. Det består grovt sett av en process som lyssnar mot FRA:n samt ett huvudprogram som har olika mus-styrda menyer genom vilka användaren kan välja olika funktioner.

- Importerade moduler:

Exec:	Innehåller rutiner för operativsystemanrop. Logitech.
Graphics:	EGA-grafikpaket, fönster, musrutiner, in/utmatning. Reglerteknik, LTH.
RTMouse:	Startar och sammanbinder mus, realtidskärna och grafik. Reglerteknik, LTH.
RealConversions:	Omvandling mellan strängar och REALs. Logitech.
Kernel:	Realtidsfaciliteter. Reglerteknik, LTH.
NumberConversion:	Omvandling heltal strängar. Reglerteknik, LTH.
SerialBuffers:	Kommunikation genom den seriella porten. Reglerteknik, LTH.
Screen:	Utskrift direkt via BIOSet. Reglerteknik, LTH.
FileSystem:	Filhanteringsrutiner. Logitech.
Strings:	Stränghantering. Reglerteknik, LTH.
MathLib:	Matematik. Reglerteknik, LTH.

Koden är uppbyggt kring två processer, huvudprogram och en rad procedurer som dess förlängda arm, några globala variabler samt en och en halv lokala moduler.

Den halva lokala modulen är ett programpaket för bodediagram som fungerar för sig själv men som inte är inlagd i en modul arbetsbesparingsskäl.

- Globala variabler:

Fem av dessa är värda att förklaras här:

settings, settingcounter:

"Settings" är den sträng i vilken de aktuella satta FRA- kommandona finns. All information programmet har om FRA:n ligger alltså här. Settingcounter håller bara reda på till vilken position settings innehåller data.

show_status:

När denna är TRUE betraktas data från FRA:n som information, som skall ekas upp direkt på skärmen.

chain_on:

När denna är TRUE utförs en serie mätningar automatiskt enligt lista skriven i filen TEMP.CHN om NEW MEASUREMENT väljs.

EndFreq:

Innehåller den sista frekvensen, som mätes vid ett svep. "Measure" avgör, när mätningen är slutförd med hjälp av detta värde.

- Uppstart, huvudprogram

Huvudprogrammet i modulen "Measurement" exekveras först. Först initialiseras realtidskärna, grafik och mus. Sedan initialiseras modulens realtidsvariabler och så startas processerna "ManualStop" och "ReadBuffer". Därefter kommer man till ett antal mindre viktiga initialiseringar. I "StartUp" läser man in den parameterfil (DEFAULT.FRE), som innehåller standardinställningen av FRA:n. Dessa parametrar läggs i settings samt sänds till FRA:n också, så att den blir som man vill ha den. Nu är vi klara och kommer in i huvudmenyn, som härefter kommer att agera "huvudprogram".

- Mätning, lokala modulen "Measurement"

Den lokala modulen "Measurement" innehåller två processer och en procedur:

(* process *) PROCEDURE ReadBuffer;

Denna ligger alltid och lyssnar mot FRA:n. Om inga speciella flaggor är satta, så betraktar den mottagen information som skräp. Detta skräp tas inte omhand. (Normal-mode)

Om däremot "show_status=TRUE" så har vi nyss gett FRA:n en statusfråga, vars svar ekas direkt upp på skärmen. "Show_status" är en gemensam resurs, men genom en kort tidsfördröjning i den anropande proceduren blir det aldrig någon konflikt. (Status-mode)

Om "Measuring=TRUE" (skyddas av semaforen "Measuringsem"), så är det mätdata som kommer in. Data placeras i strängen "store" och tas om hand av proceduren "Measure". (Mät-mode)

PROCEDURE Measure;

Anropas utanför modulen när en mätning skall utföras. Startar upp frekvensanalysexperimentet genom att starta FRA:n (generator, svep och analysator) och genom att sätta processerna "ManualStop" och "ReadBuffer" i mät-mode. Mätdata tas omhand från "ReadBuffer" genom strängen "store". "Store" är en gemensam resurs, som skyddas av semaforen "BufferSem". Data placeras i filen TEMP.MEA. Frevenserna extraheras ur datamängden allteftersom de kommer in och placeras i variabeln "check_freq". När denna fått samma värde som slutfrekvensen "EndFreq" (global), så anses mätningen vara klar. Då återgår de båda processerna till normal-mode, varvid FRA:ns generator, analysator och svep stoppas.

(* process *) PROCEDURE ManualStop;

Denna process är användarens möjlighet att avbryta en pågående mätning. Mätning stoppas genom att klicka med musen. Såväl under mätning som när mätning inte pågår är processen "non busy wait".

När man trycker på en musknapp under pågående mätning, sätts "Measuring" till "FALSE" så att programmet återgår till huvudmenyn. Dessutom skickas "BK" till FRA:n, så att svepet inte fortsätter nästa gång där den blev avbruten.

När en mätning har utförts och inte stoppats manuellt, så kommer processen fortfarande att ligga och vänta på ett musklick. Processen kommer att aktiveras första gången som en musknapp pressas ned. Den gör ingen skada, och processen lägger sig i stället och väntar på att nästa mätning skall utföras.

Utanför modulen finns mera kod, som har med mätning att göra:

PROCEDURE ChainMeasure;

Läser parameterfilnamn i filen TEMP.CHN och utför mätningarna i tur och ordning. Resultaten sorteras samman till en resultatfil.

- Sändning till FRA:n

PROCEDURE Send(ch:CHAR);

Sänder tecknet "ch" till FRA:n.

PROCEDURE SendString(string:StringType;n:CARDINAL);

Skickar de n första tecknen i "string" till FRA:n. Därefter sänds "carriage return".

- Behandling av "settings"

PROCEDURE InitFRA(filename:FileNameType);

Läser en uppsättning parametrar från en fil, lägger in parametrarna i "settings", samt se till att parametrarna sänds till FRA:n.

PROCEDURE SetFRA(parameter:StringType);

Ger ett kommando till FRA:n. Parametern sparas även i "settings", så att programmet vet vad som gäller. Gamla kommandon som gäller samma sak måste tas bort.

PROCEDURE WriteSettings(filename:FileNameType);

Informationen i "settings" sparas på filen "filename".

PROCEDURE ShowSettings(bredd,hojd:REAL);

Anropas från "ManualInitMenu". Visar nuvarande inställningar på skärmen, dvs de inställningar man kan göra i denna meny.

PROCEDURE AmplValues(bredd,hojd:REAL);

Anropas från "AmplMenu" och visar innevarande värden på de parametrar, som kan ställas i "AmplMenu".

PROCEDURE ExtractValue(VAR s:StringType;i:CARDINAL);

Arbetsrutin som används av de båda procedurerna ovan. Rutinen läser det siffervärde, som kommer närmast efter positionen i "settings" och returnerar detta i "s".

PROCEDURE RemoveSettingIfExisting(inputstring:StringType);

Raderar den delsträng som ligger i "inputstring" ur "settings" om "inputstring" finns i "settings". Eventuella siffror efter delsträngen raderas också.

- Inmatningsrutiner

PROCEDURE GetValue(VAR value:StringType;
numbertype: Numbertype);

Inmatningsrutin för en "cardinal" eller "real".

PROCEDURE GetFileName(VAR filename:FileNameType;
extention:extentiontype);

Inmatningsrutin för filnamn. Endast bokstäver och siffror accepteras. Användaren kan inte välja "extention", eftersom programmet vill att alla filer av samma slag skall ha samma "extention". Endast de 8 första godkända tecknen tas således om hand. Filerna kan vara av typerna .MEA, .FRA eller .CHN.

- Menyer

En rad mus-menyer används för kommunikation med användaren. De har alla likartad uppbyggnad. Endast HMeny (huvudmenyn) skiljer sig något.

Uppbyggnaden för samtliga menyer kan gås igenom tillsammans: Inledningsvis skrivs ledtexterna ut. Första gången "HMeny" genomlöps aktiveras en rad med musrektanglar som sedan alla menyer använder sig av. Kring de av dessa rektanglar som den aktuella menyn sedan önskar använda ritas en röd ram. Markören visas och programmet ställer sig att vänta på att användaren klickar med musen i en av de röda rutorna. Varje ruta är associerad med ett nummer. Med ledning av detta nummer väljs rätt alternativ i en "CASE" alternativt en "IF-ELSIF" konstruktion.

PROCEDURE HMeny(VAR finished:BOOLEAN);
PROCEDURE ManualInitMenu(bredd,hojd:REAL);
PROCEDURE AmplMenu(bredd,hojd:REAL);
PROCEDURE StatusHandler(hojd,bredd:REAL);
PROCEDURE SaveLoadParameters(hojd,bredd:REAL);
PROCEDURE SaveLoadData(hojd,bredd:REAL);
PROCEDURE ChainMenu(bredd,hojd:REAL);

- Bodediagram

Diagrammet plottas av de data som ligger i arbetsfilen TEMP.MEA. Proceduren "Bodediagram" kan därför anropas utifrån utan några parametrar.

PROCEDURE BodeDiagram;

Skalor och skalningar görs av Scale. TEMP.MEA arbetas igenom två gånger. Första gången läses frekvens och amplitud, andra gången frekvens och fas. En punkt avsätts för varje mätpunkt, ampitud och fas i samma diagram. När diagrammet är klart, inväntas ett musklick, varefter man kommer tillbaka till huvudmenyn.

PROCEDURE HorizontalFreqPos(—);
Ger x-koordinat till skärmen av given frekvens.

PROCEDURE VerticalAmpPos(—);
Ger y-koordinat till skärmen av given amlitud.

PROCEDURE VerticalPhasePos(—);
Ger y-koordinat till skärmen av given fas.

PROCEDURE FileMaxMin(—);
Hittar max och minvärden av frekvens, amplitud och fas i en fil.

PROCEDURE CutEven(—);
Justerar max- och minvärdena så att amplitud- och frekvensaxlarna blir graderade i hela dekader. Fasaxeln justeras till jämna moduler av 90 grader.

PROCEDURE EndTest(ch:CHAR;VAR endstate:CARDINAL);
Hjälper till att hålla reda på om man håller på med läsning av ett tal, är mellan två tal eller om filen har tagit slut.

PROCEDURE OkayTest(—);
Felutskrift med radidentifiering om en konvertering mellan sträng och tal misslyckas.

PROCEDURE Scale;
Plottar diagrammets axlar, skalor samt skriver ut max- och minvärden.

- Sortering

PROCEDURE SortTogether(into,from:FileNameType);
Två datafiler sorteras ihop i frekvensordning till en tredje fil. De två källfilerna är TEMP.MEA och "from". Resultatfilen är "into".

-Appendix 3 PC-FRA, MANUAL

0. INLEDNING

Viss kunskap om Solartron 1250 är bra för att kunna tillgodogöra sig denna manual. Solartronmanualen rekommenderas, men är inte nödvändig för lite enklare analysexperiment. Hård-och mjukvarulistor finns i Appendix 1.

0.1 KOPPLINGAR

Dator-Frekvensanalysator:
Se till så att FRA:n är ordentligt inkopplad till datorn via den seriella porten.

Frekvensanalysator-Process:
På framsidan av FRA:n finns tre par in/utgångar:

- Generatorns "High" skall kopplas till processens insignal. "Low" ansluts till jord.
- Kanal 1 skall mäta insignalen till processen och kopplas därför samman med generatorn "Low" till "Low" och "High" till "High".
- Kanal 2 skall mäta processens utsignal. "Low" jordas och "High" ansluts till processens utsignal.

Alla portarna är på framsidan av FRA:n. Denna koppling dividerar utsignal med insignal, och som mätresulat fås överföringsfunktionens värde vid den aktuella frekvensen. Slå på strömmen till datorn, skriv sedan PCFRA för uppstart av systemet.

1. INSTRUKTIONER

En default-parameteruppsättning finns lagrad på filen DEFAULT.FRE, och denna läses alltid in till programmet vid uppstart samt sändes till FRA:n, så att den blir korrekt initialiserad. DEFAULT.FRE kan endast ändras med en editor. Detta skall inte göras av någon som inte är väl förtrogen med frekvensanalysatorn.

Efter initialiseringen kommer vi till huvudmenyn, MAIN MENU. Här kan 10 olika val göras genom att klicka med musen i en av de 10 rödramade rutorna. En del av valen leder till undermenyer med samma uppbyggnad, andra såsom HELP utförs direkt.

Rutorna är uppdelade i följande fyra grupper:

1.1 INITIALISE

Olika möjligheter att ställa och kontrollera FRA-parametrar .

MANUAL INITIALISATION:

Detta val ger en undermeny, där det är möjligt att ställa enskilda parametrar till FRA:n. Se punkt 2. Här kan även parametrarnas inställningar ses.

SAVE/LOAD PARAMETERS:

Den fördefinierade parameteruppsättningen som tidigare lästs in i programmet tillsammans med de eventella ändringar, som gjorts under rubriken manual initialisation, finns lagrat internt i programmet. Denna nya parameteruppsättning kan sparas på en egen fil och senare läsas in på nytt. Se punkt 3.

CHAIN:

Ger möjlighet att koppla samman en rad parameteruppsättningar, så att en hel serie med mätningar utförs i rad. Se punkt 4.

STATUS:

En meny där samtliga parametrars värden kan kontrolleras. Se punkt 5.

1.2 MEASURE

Ny mätning eller addition av datafiler.

NEW MEASUREMENT:

Ett frekvensanalysexperiment utförs enligt de parametrar som är fördefinierade. Resultatet skrivs på filen TEMP.MEA. Mätningen startar omedelbart vid klick i denna ruta. Är CHAIN satt till ON utförs en multipel mätning, enligt den specifikation som finns given i en särskild fil. Mätningen kan avbrytas genom att klicka med musen under pågående mätning. Hittills mottagna mätdata finns i TEMP.MEA, men sista raden kan vara ofullständig, varför det rekommenderas att putsa filen i en editor, om den skall användas vid identifiering.

ADD DATA FILES:

Aktuell mätdatafil (TEMP.MEA) adderas till en tidigare undansparad mätdatafil. Detta val ger först en förteckning över undansparade mätdatafiler. Sedan ombeds man att ange ett av de existerande filnamnen, varefter filerna sorteras samman under det valda filnamnet. TEMP.MEA förblir intakt.

1.3 RESULT OUTPUT

Spara/hämta data, Bodediagram.

SAVE/LOAD MEASUREMENT DATA:

Läsa in gamla data eller spara aktulla data på egen fil. Se punkt 6.

BODE DIAGRAM:

Bodediagram ritas upp av de data som ligger i filen TEMP.MEA. Önskas Bodediagram på någon annan datafil än nämnda, måste denna först läsas in i programmet enligt punkt 6.

Uppbyggnad:

Amplituden: Vänster lodräta axel är logaritmisk och graderad i decilog, dvs 10*log x. Dess färg är ljusröd och tillhörande mätpunkter röda. Skalan är indelad i jämna dekader med max och minvärde angivna. Omfattar skalan mer än en dekad, utmärks dekadgränsen med en lång pinne, de kortare pinnarna utgörs av mellanvärdena 2 och 5 i linjär skala.

Frekvensen: Den horisontella axeln är en logaritmerad frekvensaxel, som är gemensam för både amplitud och fas. Max- och minvärden är angivna.

Fasen : Höger lodräta axel är linjär och graderad i jämna 90-graders moduler. Axeln är brun, dess associerade mätpunkter gula. Max- och minvärden är utsatta, samt en pinne för mellanliggande modulgränser.

Klicka med musen för att komma tillbaka till huvudmenyn.

För mera avancerad analys rekommenderas PC-MATLAB.

1.4 Övrigt

Hjälp, tillbaka till DOS.

HELP:

Hjälpfunktion för huvudmenyn. Nytt klick med musen gör att man kommer tillbaka till huvudmenyn.

BACK TO DOS:

Lämna programmet. Den enda information som kan förloras är ändringar i parameteruppsättning som ej sparats. Varning härför utfärdas inte.

En del av valen leder till undermenyer med samma typ av uppbyggnad, andra såsom HELP utförs direkt.

Även mätresultat som inte explicit sparats ligger på editerbar fil. Data förloras således inte, om man går ur programmet eller slår av datorn.

2. INSTÄLLNING AV FREKVENSANALYSATORN

Under rubriken MANUAL INITIALISATION kan man göra de inställningar av frekvensanalysatorn, som har bedömts vara användbara. Skulle någon gång

andra kommandon behöva ges, får dessa skrivas in i en parameterfil, som man sedan läser in till programmet. Se punkt 3.

FRA:n kan delas upp i tre avdelningar: Generator, analysator och svep. Dessa behandlas nedan var för sig. Samtliga val göres på samma sätt som i huvudmenyn: Klicka i önskad ruta med musen.

2.1 GENERATORN

Generatorn är den enhet som skickar ut insignalen till processen. De olika inställningsmöjligheterna följer här:

AMPLITUDE (vald i manual initialisation menyn):

Här fås en undermeny med inställningsmöjligheter:

AMPLITUDE (vald i amplitude menyn):

Här ställes generatorns amplitud. Gäller om AMPLITUDE COMPRESSION är OFF, annars saknar den funktion.

AMPLITUDE COMPRESSION ON/OFF:

Innebär helt enkelt att man väljer om man vill ha amplitude compression eller inte.

Följande är amplitudinställning när COMPRESSION är ON: SOURCE är den kanal som amplituden skall ut på. VALUE är den amplitud som skall försöka hållas. ERROR är den procentram inom vilken amplituden skall försöka hållas. Om FRA:n piper och visar ERROR 84 (inte datorn alltså), så klarar den inte att hålla amplituden med givna felgränser. Se FRA:ns manual. Ofta räcker det med att öka ERROR något för att genomföra experimentet.

BIAS:

Ange värdet på biasen.

WAVEFORM:

Ger dig möjligheterna sinus eller fyrkantvåg.

2.2 SVEPET (SWEEP)

Programmet tillåter svep över frekvensintervall. Svepet är alltid inkopplat.

MIN FREQUENCY:

Den frekvens i Hz anges vid vilken svepet skall börja.

MAX FREQUENCY:

Den frekvens anges vid vilken svepet upphör.

LOG INCREMENT:

Endast logaritmisk stegning anses för våra ändamål vara användbar. Två typer finns (fås i en ruta bredvid):

-STEPS/DECADE: Anger hur många mätningar vi vill ha per frekvensdekad.
-STEPS/SWEEP : Anger hur många mätningar vi vill ha per svep.

2.3 ANALYSATORN (ANALYSER):

Analysatorn kan ställas in med fördröjning och integration.

MEASUREMENT DELAY:

I en ruta bredvid kan mätfördröjning väljas i sekunder eller cykler.

INTEGRATION:

Också integrationen kan väljas i sekunder eller antalet cykler i en ruta bredvid. Dessutom finns det AUTO INTEGRATION, som kan vara fördelaktig, eftersom detta innebär att FRA:n endast vid behov använder långa integrationer. Val av kanalnummer samt LONG eller SHORT autointegrationstid finns här. Slutligen finns även HELP samt BACK TO PREVIOUS MENU här.

3. SPARA OCH HÄMTA PARAMETERUPPSÄTTNING

Valet SAVE/LOAD PARAMETERS ger en ny menu med dessa möjligheter:

SAVE :

Alla inställningar man gjort kan sparas på editerbar fil. Samma mätning kan då lätt upprepas vid ett senare tillfälle. Annars har man möjlighet att med en editor gå in här och lägga in speciella inställningar, som programmet inte stöder. Klicka i SAVE så visas samtliga filer *.FRA. Mata därefter in ett filnamn om högst 8 tecken, varefter parameteruppsättningen kommer att sparas under valt filnamn. Filnamnets extension kan man inte välja, utan den blir alltid .FRA.

LOAD :

Samtliga filer *.FRA visas, varefter Du ombeds mata in namnet för en av dessa existerande parameterfiler. Denna uppsättning parametrar läses därefter in i programmet och transmitteras till FRA:n.
Även HELP och BACK TO MAIN MENU finns, liksom i andra undermenyer.

4. CHAIN:

Om man i ett frekvensanalysexperiment skulle behöva olika parameteruppsättningar för olika frekvensområden, så är CHAIN användbart. (T.ex. om man vill kunna mäta lite tätare i något speciellt intressant frekvensområde.) CHAIN innebär att man på en fil med extension .CHN lagrar filnamnen på ett antal parameteruppsättningsfiler (d.v.s. filer med extension .FRA, se punkt 3.). Den stora fördelen med CHAIN blir att man får hela mätserien gjord utan att behöva sitta och passa delmätningar, som kan ta lång tid. Om man väljer CHAIN i huvudmenyn, så kommer man ner till en undermeny:

ON/OFF:

"Toggle"-inställning av CHAIN. Är den ON blir den alltså OFF och vice versa. Om ON så kommer en NEW MEASUREMENT i huvudmenyn att bli en kedjemätning enligt den lista av parameteruppsättningar, som finns i filen TEMP.CHN. Om OFF begäres, så utförs mätningen som vanligt.

MAKE CHAIN:

Göra en ny CHAIN-lista. Lista över befintliga parameterfiler *.FRA visas på skärmen. Sedan frågar programmet omväxlande om filnamn, och huruvida man önskar fortsätta. Efter svaret "N" eller "n" (står för "nej") är CHAIN-listan klar och skriven på filen TEMP.CHN.

SAVE CHAIN:

Redan befintliga CHN-filer visas. Sparar data som finns i TEMP.CHN under eget filnamn.CHN. "SAVE CHAIN" döper om TEMP.CHN, varför denna fil inte kommer att existera efter det att "save" har gjorts. Du kan inte själv välja extension, utan den är alltid satt till .CHN.

LOAD CHAIN:

Filerna *.CHN visas på skärmen. Du ombeds välja ett av de existerande filerna, som därefter kopieras in på arbetsfilen TEMP.CHN .

SEE CHAIN:

Innehållet i TEMP.CHN visas på skärmen. Tryck på en musknapp, när Du har tittat färdigt.

Behöver någon CHN-fil en ändring, måste den skrivas om helt eller ändras utanför programmet i en editor.

5. STATUS

Programmet stöder ju inte alla FRA:ns möjligheter. Om man vill se någon speciell inställning, så kan man göra det här. Det är också så att de inställningar som visas under manual initialisation är så som programmet tror att det är. I STATUS visas de kompletta parameterinställningarna direkt från FRA:n. Det finns ett antal underrubriker, men de talar för sig själv.

6. SAVE/LOAD MEASUREMENT DATA

Resultatet från den senast gjorda eller hämtade mätningen finns i den editerbara filen TEMP.MEA.

SAVE DATA:

Redan befintliga MEA-filer visas. Detta kommando sparar data som finns i TEMP.MEA under eget filnamn.MEA. Detta kommando döper om TEMP-.MEA, varför denna fil inte kommer att existera efter det att "save" har gjorts. Du kan inte själv välja filnamnets extension, utan den är alltid satt till .MEA.

LOAD DATA:

Filerna *.MEA visas på skärmen. Du ombeds välja en av de existerande filerna, som därefter kopieras in på arbetsfilen TEMP.MEA .

7. FILSTRUKTURER

Tre typer av filer finns:

*.FRA	innehåller parameteruppsättning för FRA:n.
*.CHN	innehåller en följd av filnamn av typen ovan. Denna uppräkning av filnamn möjliggör multipla mätningar.
*.MEA	innehåller mätdata.

Samtliga är editerbara i en vanlig editor.

7.1 .FRA

Filer av typen FRA är uppbyggda enligt:

kommando semikolon kommando semikolon osv.

Exempel:

GD1;
BI0.5;
AE5.0;
#

Semikolonet tolkas av programmet som att det skall sända iväg kommandot. Filen kan lika gärna vara en sträng i stil med

GD1;BI0.5;AE5.0;

Utformningen kan Du välja själv om Du editerar filen i en extern editor. Programmet skriver alltid på den sista formen. Parametrarnas innebörd finns förklarade i FRA- manualen, sidorna 13.9-13.18.

7.2 .CHN

Filen består av en kolumn av filnamn av typen *.FRA.

Exempel:
LOWFREQ .FRA
MIDDELFR.FRA
HIFREQ .FRA
#

Skiljetecken är EOL.

7.3 .MEA

Detta är våra mätdatafiler. De har tre kolumner per rad. Första kolumnen innehåller frekvenserna, den andra innehåller amplituderna och den tredje fasen i grader. Varje kolumn består av en "real" med strukturen

<tecken><5-siffrig mantissa><E><tecken><2-siffrig exponent>

Exempel:

+2.0000E+01 +2.2626E-03 −2.0403E+01
+2.5778E+01 +1.0621E-03 −1.2940E+02
+3.1698E+01 +6.7591E-03 +7.3163E+01
+3.5718E+01 +1.0855E-03 +1.1258E+01
#

OBS

Varje tal har 11 tecken. Kolumnerna åtskiljs med två blanktecken. Editerar Du filen, se då till att talen blir just 11 tecken långa, samt att det inte finns fler än 2 blanka mellan kolumnerna. Programmet uppfattar nämligen fler än 2 blanka som filslut. De två första kolumnerna får inte innehålla negativa tal.

8. STANDARDISERINGAR

Flera parametrar som inte kan sättas inuti programmet är satta till standardvärden i default-filen DEFAULT.FRE. Filen har samma uppbyggnad som beskrives i (7.1). Här följer hela filen med kommentarer. (M 3.3) betyder hänvisning till Solartronmanualen sidan 3.3.

AM0.5; Amplitud

BI0.0; Ingen bias

WF0; Vågform sinus mest användbar

IC5; Analysatorns integrationstid 5 cykler

AU0; Auto integration off (M 5.4)

MS5.0; Tidsfördröjning (M 5.3)

HA1; "Analysers harmonic 1"= mäter på just den frekvens vi har angett, inte på några högre ordningars svängningar

MA40.0; Max frekvens (Hz)

MI0.5; Min frekvens (Hz)

GD1E1; Svepet ökar frekvensen logaritmiskt med 10 mätningar per dekad.

RA1,4; Kanal 1 max 30 volt

RA2,4; Kanal 2 max 30 volt

DC1,0; DC-koppling kanal 1

DC2,0; DC-koppling kanal 2. (M 5.6)

IP1,0; Kanal 1 inkopplad i fronten

IP2,0; Kanal 2 inkopplad i fronten

OP1,1; Alla data ut på RS423-porten (ändra ej)

CO1; Koordinater i amplitud och fas (ändra ej)

AP0; Amplitude compression off. (M 9.5)

AE10; Felmarginal 10

AS100; Kanal 1 som källa (source)

AV0.5; Försök hålla amplituden till 0.5 Volt.

SO0201; Kanal 1 mäter på insignalen, kanal 2 mäter utsignalen. Mätdata blir sedan Ch2/Ch1, dvs överföringsfunktionens värde vid den aktuella frekvensen.

***

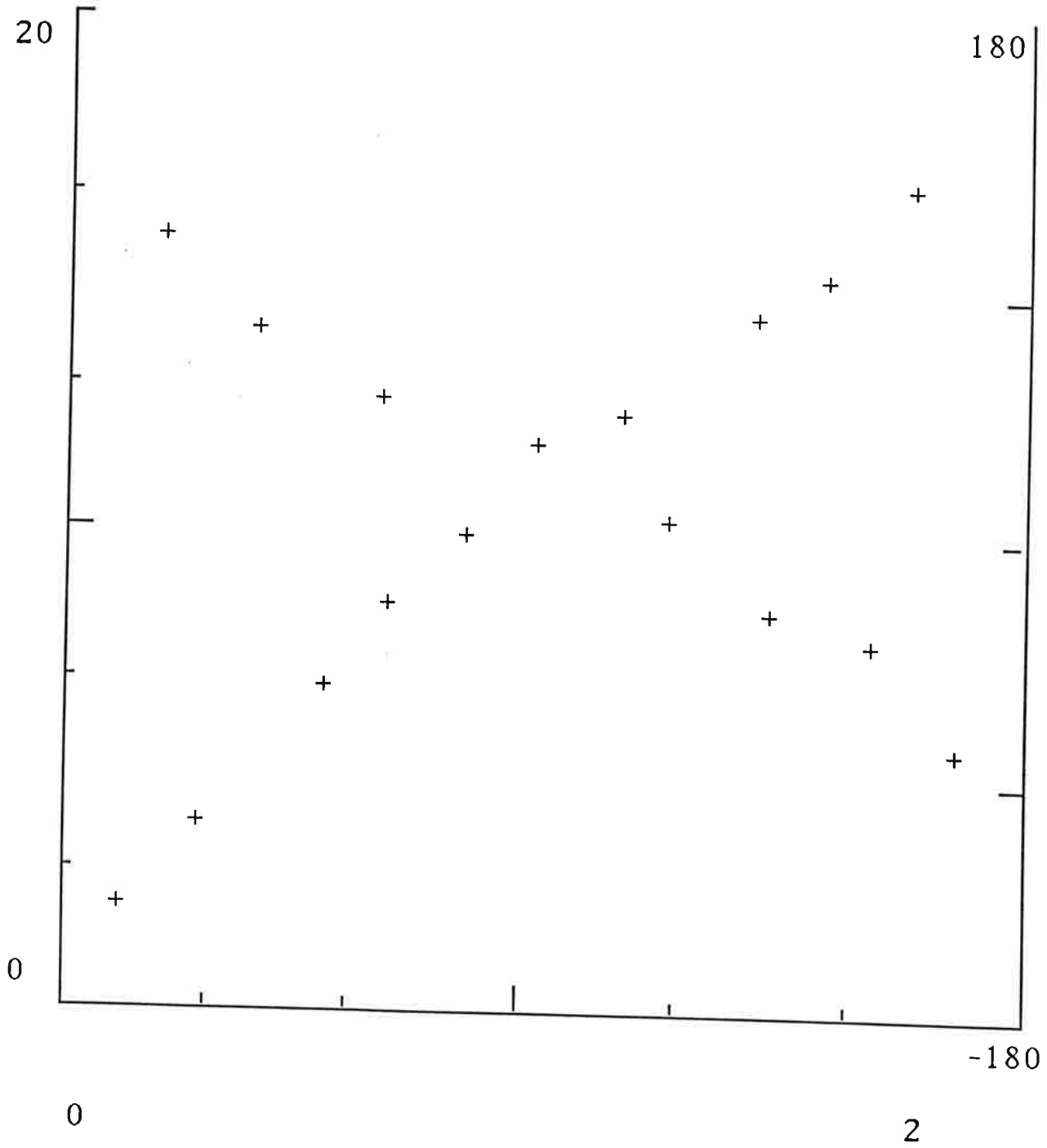
- APPENDIX 4
SKÄRMBILDER

I detta appendix finns alla skärmlayouter samlade tillsammans med några förklarande figurer.

1. Förklaring av menyernas uppbyggnad.
2. Huvudmenyn.
3. Menyn för manuell initialisering.
4. Exempel på val/inmatning av data i en meny.
5. Menyn för inställing av amplitud.
6. Menyn för att spara och hämta parameteruppsättningar.
7. Menyn för att spara och hämta mätdata.
8. Statusmenyn.
9. Exempel/förklaring av bodediagrammet.

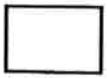
EXEMPEL PÅ HUR BODEDIAGRAMMET KAN SE UT.

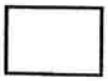
Vänstra vertikala axeln är amplitudaxeln graderad i decilog, dvs10 log x. Den har i programmet färgen ljusröd och tillhörande mätpunkter är röda kryss.
Den horisontella axeln är i programmet vit. Denna axel är frekvensaxeln. Skalan är logaritmerad.
Högra vertikala axeln är fasaxeln. Den är linjär och graderad i grader. Färgen är brun och de tillhörade mätpunkerna är gula.

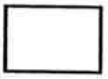


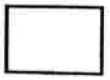
STATUS


GENERATOR

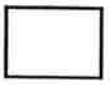

ANALYSERS

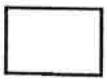

SWEEP

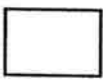

DISPLAY


PLOT

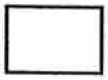

SYNC


PROGRAM

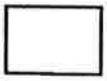

HELP


BACK TO MAIN MENU

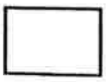
SAVE/LOAD DATA



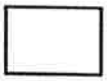
LOAD DATA



SAVE DATA

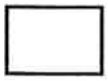


HELP



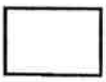
BACK TO MAIN MENU

SAVE/LOAD PARAMETERS

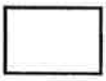


LOAD DATA

SAVE DATA



HELP



BACK TO MAIN MENU

AMPLITUDE

COMPR OFF: ☐ AMPLITUDE 0.5 V

☐ AMPLITUDE COMPRESSION ON/OFF OFF

COMPR ON: ☐ SOURCE Channel 1

☐ VALUE 0.5 V

☐ ERROR 10 %

☐ HELP

☐ BACK TO PREVIOUS MENU

MANUAL INITIALISATION

GENERATOR:	☐	AMPLITUDE	0.5 V
	■	BIAS	0.0 V
	☐	WAVEFORM	SINUS
SWEEP:	☐	MIN FREQUENCY	0.05 Hz
	☐	MAX FREQUENCY	50 Hz
	☐	LOG INCREMENT	10 STEPS/DECADE
ANALYSER:	☐	INTEGRATION TIME	5 CYCLES
	☐	MEASUREMENT DELAY	5 s

SETTING OF BIAS
NEW VALUE:

☐ HELP

☐ BACK TO MAIN MENU

MANUAL INITIALISATION

GENERATOR:	☐	AMPLITUDE	0.5 V
	☐	BIAS	0.0 V
	☐	WAVEFORM	SINUS
SWEEP:	☐	MIN FREQUENCY	0.05 Hz
	☐	MAX FREQUENCY	50 Hz
	☐	LOG INCREMENT	10 STEPS/DECADE
ANALYSER:	☐	INTEGRATION TIME	5 CYCLES
	☐	MEASUREMENT DELAY	5 s

☐ HELP

☐ BACK TO MAIN MENU

MAIN MENU

INITIALISE: ☐ MANUAL INITIALISATION

☐ CHAIN IS OFF

☐ SAVE/LOAD PARAMETERS

☐ STATUS

MEASURE: ☐ NEW MEASUREMENT

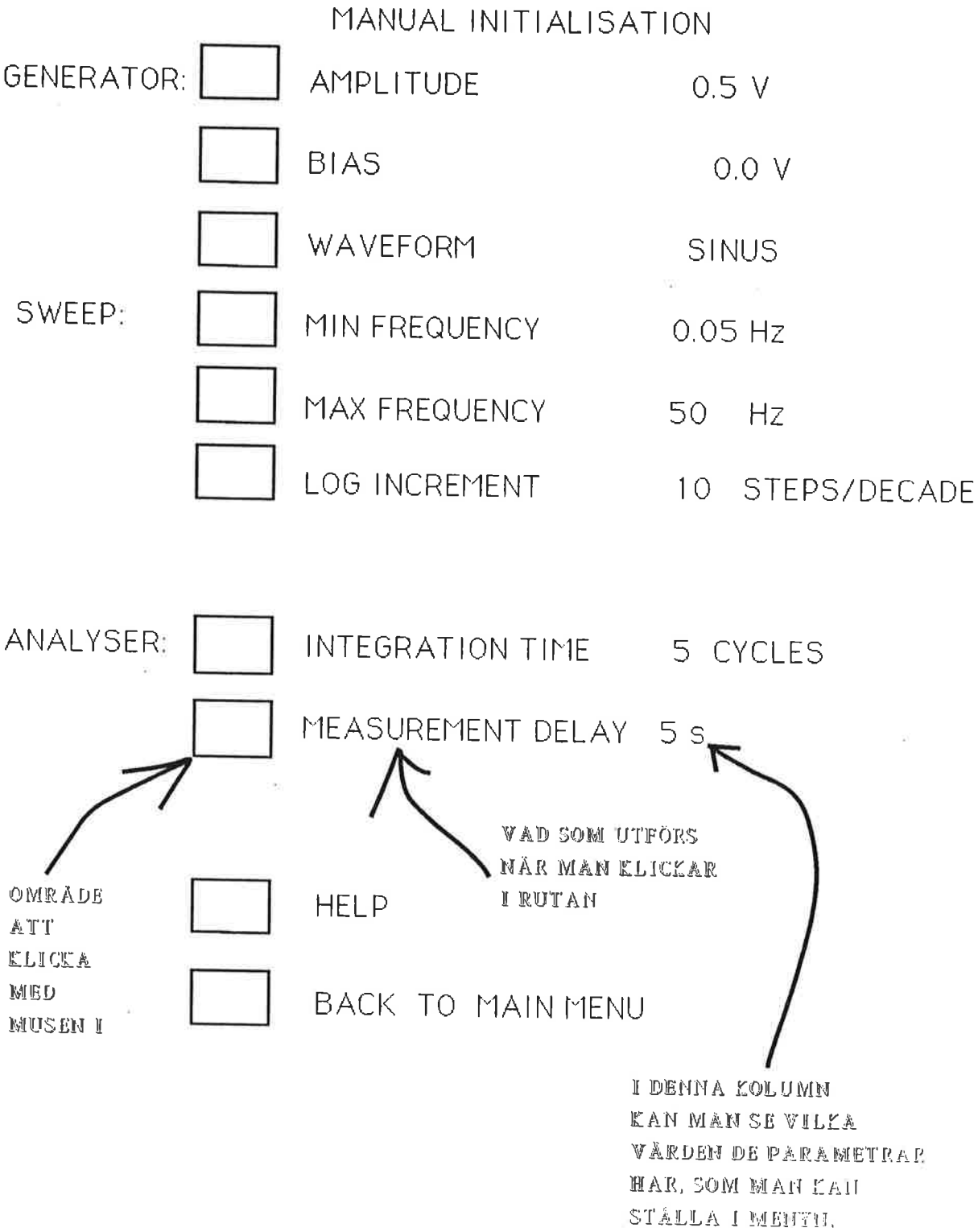
☐ ADD DATA FILES

RESULT- ☐ SAVE/LOAD MEASUREMENT DATA

OUTPUT: ☐ BODE DIAGRAM

☐ HELP

☐ BACK TO DOS



MENYUPPBYGGNAD

* REFERENSER

- Boris Magnusson - Realtidsprogrammering föreläsningsanteckningar
  Inst för Datateknik Datalogi och Numerisk Analys, Lund 1986.
  Förklarar realtidsprogrammering. Innehåller även en kortfattad beskrivning av Modula-2.

- Eric Astor - Från Pascal till Modula-2
  Studentlitteratur 1986.
  Bra, komplett beskrivning av programspråket Modula-2.

- 1250 Frequency Response Analyser, Operating Manual
  Solartron Instrumentation Group, april 1983.
  Orginalmanual till frekvensanalysatorn.

- 1250 FREQUENCY RESPONSE ANALYSER IBM-PC INTERFACING VIA GPIB
  Solartron Instruments, Reference P/1250/01
  Alternativ till mitt program.

- Leif Andersson - Report about communication...
  Institutionen för Reglerteknik, LTH 1988
  Beskrivning av modulen SerialBuffers m.m.

- BILAGA 1 PROGRAMLISTNING

```
MODULE PCFRA;

(* ********************************************************************** *)
(* PC-FRA v1.0                                                            *)
(* Control program for Solarton 1250 Frequency Response Analyser          *)
(* Written by Eric Wichtel 1989                                           *)
(* under supervision from DrSc Rolf Johansson and MSc Leif Andersson      *)
(* ********************************************************************** *)
(* Department of Automatic Control  *  Telephone: +46 46 107000           *)
(* Lund Institute of Technology     *  Telex    : S-33533 LUNIVER         *)
(* Box 118                          *  Telefax  : +46 46 138118           *)
(* S-221 00 LUND                    *                                     *)
(* Sweden                           *                                     *)
(* ********************************************************************** *)

FROM Exec IMPORT DosCommand;
FROM Graphics IMPORT
     handle,rectangle,color,VirtualScreen,SetWindow,SetViewPort,WriteString,
     DrawRectangle,SetMouseRectangle,WaitMouseRectangle,CharacterSize,point,
     SetTextColor,SetLineColor,ReadString,ShowCursor,HideCursor,FillRectangle,
     SetFillColor,EraseChar,buttonset,buttontype,
     WaitForMouse,PolyMarker,PolyLine;
IMPORT RTMouse;
FROM RealConversions IMPORT StringToReal,RealToString;
FROM Kernel IMPORT CreateProcess,WaitTime,SetPriority,InitSem,Wait,Signal,
     InitEvent,Await,Semaphore,Event,Cause;
FROM NumberConversion IMPORT CardToString,StringToCard;
IMPORT SerialBuffers;
FROM SerialBuffers IMPORT BufferPointer, TransmitBuffer, ReceiveBuffer;
FROM Screen IMPORT Write;
FROM FileSystem IMPORT Delete,Again,Lookup,Reset,ReadChar,WriteChar,
     Close,File,Rename;
FROM Strings IMPORT Pos,Insert;
FROM MathLib IMPORT ln;
CONST EOL=36C;
      EOF=32C;
TYPE StringType=ARRAY [0..11] OF CHAR;
     FileNameType=ARRAY [1..12] OF CHAR;
     NumberType=(cardinal,real);
     extentiontype=(FRA,MEA,CHN);
VAR  TransmitBuf, DisplayBuf: BufferPointer;
     string:StringType;
     settings:ARRAY [0..200] OF CHAR;     (* global variabel dar alla FRA *)
     settingcounter:CARDINAL;             (* variabler finns lagrade      *)
     finished:BOOLEAN;                    (* tillbaka till dos            *)
     show_status:BOOLEAN;                 (* Om show_status=TRUE saa skall
                                             alla tecken fraan FRA:n ekas
                                             direkt ut paa skaermen       *)
     chain_on:BOOLEAN;  (*  if true then utfoers en serie av maetningar
                            enligt listan i filen TEMP.CHN, om man
                            aktiverar New Measurerment eller Measurement
                            To Add To Previous Measurement *)
     h:handle;
     rmax:rectangle;
     p:point;
     EndFreq:REAL;  (* contains a measurement's last frequency *)

MODULE Measurement;
IMPORT CreateProcess,DisplayBuf,WaitTime,SetPriority,InitSem,Wait,Signal,
InitEvent,Await,Semaphore,Event,Cause,FillRectangle,rmax,
Lookup,Reset,ReadChar,WriteChar,Close,File,SerialBuffers,TransmitBuffer,
StringToReal,BufferPointer,ReceiveBuffer,TransmitBuf,
buttonset,WaitForMouse,point,WriteString,RTMouse,show_status,
SendString,h,Write,EndFreq,DosCommand,EOL;
EXPORT Measure;

CONST store_size=1000;
TYPE StringType=ARRAY [0..11] OF CHAR;
VAR store:ARRAY[1..store_size] OF CHAR;
    j:CARDINAL;   (* last element in store *)
```

```
    MeasurementStart,BufferSem,MeasuringSem,do_frcheck,frcheck_done:Semaphore;
    data_arrived:Event;
    Measuring:BOOLEAN;

(* process *) PROCEDURE ReadBuffer;
(* this process reads what comes from the FRA, and sends it to *)
(* Measure, ignores it or writes it directly on the screen. *)
VAR i:CARDINAL;
    ch:CHAR;
BEGIN
  SetPriority(20);
  j:=0;
  LOOP
    ReceiveBuffer(DisplayBuf);
    Wait(BufferSem);
    WITH DisplayBuf^ DO
      FOR i:=0 TO nx-1 DO
        ch:=text[i];
        IF show_status THEN  (* show_status is set in HMeny *)
           Write(ch);
        END;
        Wait(MeasuringSem);
        IF Measuring THEN   (* if not the character is not used *)
             INC(j);
             store[j]:=ch;
        END;
        Signal(MeasuringSem);
      END;
    END;
    Cause(data_arrived);
    Signal(BufferSem);
  END;
END ReadBuffer;



(* process *) PROCEDURE ManualStop;
VAR b:buttonset;
    p:point;
BEGIN
    SetPriority(20);
    LOOP
      Wait(MeasurementStart);
      WaitForMouse(h,p,b);
      Wait(MeasuringSem);
        IF Measuring THEN
           SendString("BK",2);  (* break in *)
        END;
        Measuring:=FALSE;
      Signal(MeasuringSem);
    END;
END ManualStop;



PROCEDURE Measure;
(* starta maetning *)
VAR i,k: CARDINAL; ch: CHAR;  (* These are just temporary variables. *)
    p:point;
    fil:File;           (* initialized file structure *)
    status,okay,last_line:BOOLEAN;
    check:StringType;
    check_freq:REAL;
BEGIN
  FOR i:=1 TO store_size DO
      store[i]:=CHR(0);
  END;
  DosCommand('del','temp.mea',okay);
  FillRectangle(h,rmax);
  p.h:=0.3;
  p.v:=0.6;
```

```
  WriteString(h,p,"             TAKING MEASUREMENTS...");
  p.v:=0.5;
  WriteString(h,p,"STOP GENERATOR/MEASUREMENT BY PRESSING MOUSE BUTTON");
  CreateProcess(ManualStop,1000);
  SendString("SC1",3);  (* start sweep log up *)
  SendString("RE",2);   (* recycle *)
  SendString("RG",2);   (* start generator *)
  i:=1;
  Lookup(fil,"temp.mea",TRUE);
  Reset(fil);
  last_line:=FALSE;
  Wait(MeasuringSem);
    Measuring:=TRUE;
  Signal(MeasuringSem);
  Signal(MeasurementStart);
  REPEAT
    REPEAT
      Wait(BufferSem);
      IF (store[i]="+") OR (store[i]="-") THEN
          FOR k:=0 TO 11 DO
              check[k]:=CHR(0);
          END;
          FOR k:=1 TO 37 DO
              WriteChar(fil,store[i]);
              IF k<12 THEN
                 check[k-1]:=store[i];
              ELSIF k=12 THEN
                 StringToReal(check,check_freq,okay);
                 IF check_freq=EndFreq THEN
                    last_line:=TRUE;
                 END;
              END;
              INC(i);
              Wait(MeasuringSem);
               status:=Measuring;
              Signal(MeasuringSem);
              WHILE (i>j) AND status DO
                Await(data_arrived);
                Wait(MeasuringSem);
                 status:=Measuring;
                Signal(MeasuringSem);
              END;
          END;
          WriteChar(fil,EOL);
      ELSIF last_line AND (store[i]='Y') THEN
          Wait(MeasuringSem);
          Measuring:=FALSE;
          Signal(MeasuringSem);
          INC(i);
      ELSE
          INC(i);
      END;
      Signal(BufferSem);
      Wait(MeasuringSem);
        status:=Measuring;
      Signal(MeasuringSem);
    UNTIL (i>j) OR NOT status;
    i:=1;
    j:=1;
  UNTIL NOT status;
  WriteChar(fil,CHR(32B));              (* end of file *)
  Close(fil);
  SendString("SQO",3);  (* stop generator *)
  SendString("SS",2);   (* stop sweep *)
  SendString("SA",2);   (* stop analysers *)
END Measure;

BEGIN
  show_status:=FALSE;
  RTMouse.Init;
  InitSem(BufferSem,1);
```

```
  InitSem(MeasuringSem,1);
  InitSem(do_frcheck,0);
  InitSem(MeasurementStart,0);
  InitSem(frcheck_done,0);
  InitEvent(data_arrived,BufferSem);
  SerialBuffers.Init(TransmitBuf);
  Wait(MeasuringSem);
     Measuring:=FALSE;
  Signal(MeasuringSem);
  CreateProcess(ManualStop,1000);
  CreateProcess(ReadBuffer,5000);
END Measurement; (* local module *)


PROCEDURE GetFileName(VAR filename:FileNameType;extention:extentiontype);
(* inmatningsrutin for filnamn *)
VAR p:point;                    (* cursorns lage *)
    workstring:StringType;      (* innehaller det ofiltrerade filnamnet *)
    nrofchar:CARDINAL;          (* raknar antal filtrerade/godkanda tecken *)
    i:INTEGER;                  (* vanlig raknevariabel *)
BEGIN
  FOR i:=1 TO 8 DO
     filename[i]:=" ";
  END;
  p.h:=0.0;
  p.v:=0.15;
  SetTextColor(h,black);
  WriteString(h,p,"Filename (max 8 characters):");
  SetTextColor(h,white);
  p.v:=0.05;
  ReadString(h,p,workstring);
  i:=-1;
  nrofchar:=0;

  REPEAT  (* workstring filtreras och godkanda tecken kopieras till filename*)
    INC(i);
    IF ((workstring[i]>="0") AND (workstring[i]<="9")) OR
       ((workstring[i]>="A") AND (workstring[i]<="Z")) OR
       ((workstring[i]>="a") AND (workstring[i]<="z")) THEN
                INC(nrofchar);
                filename[nrofchar]:=workstring[i];
    END;
  UNTIL (i=11) OR (nrofchar=8) OR (ORD(workstring[i])=0);
  filename[9]:=".";
  IF extention=FRA THEN
    filename[10]:="F";
    filename[11]:="R";
    filename[12]:="A";
  ELSIF extention=MEA THEN
    filename[10]:="M";
    filename[11]:="E";
    filename[12]:="A";
  ELSE
    filename[10]:="C";
    filename[11]:="H";
    filename[12]:="N";
  END;
END GetFileName;


PROCEDURE SaveLoadData(hojd,bredd:REAL);

VAR r:rectangle;
    p:point;
    b:buttonset;
    i:CARDINAL;
    okay,menufinished:BOOLEAN;
    fil,copy_to:File;
    ch:CHAR;
    filename:FileNameType;
```

```
BEGIN
 menufinished:=FALSE;
 FillRectangle(h,rmax);
 REPEAT
   SetTextColor(h,intensewhite);
   p.h:=bredd*31.0;
   p.v:=hojd*23.0*16.0/24.0;
   WriteString(h,p,"SAVE/LOAD DATA");
   SetTextColor(h,white);
   SetLineColor(h,red);
   p.h:=18.0*bredd;
   p.v:=hojd;
   WriteString(h,p,"BACK TO MAIN MENU");
   p.v:=2.0*hojd;
   WriteString(h,p,"HELP");
   p.v:=13.0*hojd;
   WriteString(h,p,"SAVE DATA");
   p.v:=14.0*hojd;
   WriteString(h,p,"LOAD DATA");
   r.loleft.h:=11.0*bredd;
   r.upright.h:=16.0*bredd;

   r.loleft.v:=2.0*hojd;
   r.upright.v:=3.0*hojd-0.01;
   DrawRectangle(h,r);
   r.loleft.v:=hojd;
   r.upright.v:=2.0*hojd-0.01;
   DrawRectangle(h,r);
   r.loleft.v:=14.0*hojd;
   r.upright.v:=15.0*hojd-0.01;
   DrawRectangle(h,r);
   r.loleft.v:=13.0*hojd;
   r.upright.v:=14.0*hojd-0.01;
   DrawRectangle(h,r);

 ShowCursor;
 i:=WaitMouseRectangle(h);
 r.loleft.v:=FLOAT(i-1)*hojd;
 r.upright.v:=FLOAT(i)*hojd-0.01;
 SetFillColor(h,red);
 HideCursor;
 FillRectangle(h,r);
 SetFillColor(h,black);
 IF i=2 THEN
   menufinished:=TRUE;   (* back to main menu *)

 ELSIF i=3 THEN (* help function *)
    FillRectangle(h,rmax);
    p.h:=0.0;
    p.v:=0.95;
    WriteString(h,p,"HELP LOAD/SAVE DATA");
    p.h:=0.0;
    p.v:=0.7;
    WriteString(h,p,"SAVE DATA means that your data presently in the file TEMP
.MEA");
    p.h:=0.0;
    p.v:=0.6;
    WriteString(h,p,"is renamed to the filename.MEA you specify.");
    p.h:=0.0;
    p.v:=0.4;
    WriteString(h,p,"LOAD DATA means that an existing data file.MEA");
    p.h:=0.0;
    p.v:=0.3;
    WriteString(h,p,"is copied into the workfile TEMP.MEA.");
    p.h:=0.7;
    p.v:=0.1;
    WriteString(h,p,"PRESS MOUSE BUTTON TO RETURN");
    WaitForMouse(h,p,b);

  ELSIF i=14 THEN              (* SAVE *)
    FillRectangle(h,rmax);
```

```
      DosCommand('cls','',okay);
      DosCommand('dir/w','*.mea',okay);
      GetFileName(filename,MEA);
      Lookup(fil,'TEMP.MEA',TRUE);
      Rename(fil,filename);
      Close(fil);
  ELSIF i=15 THEN (* LOAD: An existing file is copied to TEMP.MEA *)
        FillRectangle(h,rmax);
        p.h:=0.0;
        p.v:=0.55;
        DosCommand('cls','',okay);
        DosCommand('dir/w','*.mea',okay);
        WriteString(h,p,'Which existing file.MEA to load?');
        GetFileName(filename,MEA);
          Lookup(fil,filename,FALSE);

                                  (* 'notdone' is returned to fil.   *)
          Lookup(copy_to,'TEMP.MEA',TRUE);
          REPEAT
            ReadChar(fil,ch);
            WriteChar(copy_to,ch);
          UNTIL ch=CHR(0);
          Close(fil);
          Close(copy_to);
  END;
  FillRectangle(h,rmax);
 UNTIL menufinished;
END SaveLoadData;


PROCEDURE SaveLoadParameters(hojd,bredd:REAL);

VAR r:rectangle;
    p:point;
    b:buttonset;
    i,j:CARDINAL;
    okay,menufinished:BOOLEAN;
    fil,copy_to:File;
    ch:CHAR;
    filename:FileNameType;
BEGIN
 menufinished:=FALSE;
 FillRectangle(h,rmax);
 REPEAT
   SetTextColor(h,intensewhite);
   p.h:=bredd*31.0;
   p.v:=hojd*23.0*16.0/24.0;
   WriteString(h,p,"SAVE/LOAD PARAMETERS");
   SetTextColor(h,white);
   SetLineColor(h,red);
   p.h:=18.0*bredd;
   p.v:=hojd;
   WriteString(h,p,"BACK TO MAIN MENU");
   p.v:=2.0*hojd;
   WriteString(h,p,"HELP");
   p.v:=13.0*hojd;
   WriteString(h,p,"SAVE PARAMETERS");
   p.v:=14.0*hojd;
   WriteString(h,p,"LOAD PARAMETERS");
   r.loleft.h:=11.0*bredd;
   r.upright.h:=16.0*bredd;

   r.loleft.v:=2.0*hojd;
   r.upright.v:=3.0*hojd-0.01;
   DrawRectangle(h,r);
   r.loleft.v:=hojd;
   r.upright.v:=2.0*hojd-0.01;
   DrawRectangle(h,r);
   r.loleft.v:=14.0*hojd;
   r.upright.v:=15.0*hojd-0.01;
   DrawRectangle(h,r);
```

```
    r.loleft.v:=13.0*hojd;
    r.upright.v:=14.0*hojd-0.01;
    DrawRectangle(h,r);

   ShowCursor;
   i:=WaitMouseRectangle(h);
   r.loleft.v:=FLOAT(i-1)*hojd;
   r.upright.v:=FLOAT(i)*hojd-0.01;
   SetFillColor(h,red);
   HideCursor;
   FillRectangle(h,r);
   SetFillColor(h,black);
   IF i=2 THEN
     menufinished:=TRUE;   (* back to main menu *)

   ELSIF i=3 THEN (* help function *)
      FillRectangle(h,rmax);
      p.h:=0.0;
      p.v:=0.95;
      WriteString(h,p,"HELP LOAD/SAVE PARMETERS");
      p.v:=0.7;
      WriteString(h,p,"SAVE PARAMETERS means that your present settings");
      p.v:=0.6;
      WriteString(h,p,"are saved to the filename.FRA you specify.");
      p.v:=0.4;
      WriteString(h,p,"LOAD PARAMETERS  means that an existing setting file.FRA"
);
      p.v:=0.3;
      WriteString(h,p,"is read into the program and transmitted to the analyser.
");
      p.h:=0.7;
      p.v:=0.1;
      WriteString(h,p,"PRESS MOUSE BUTTON TO RETURN");
      WaitForMouse(h,p,b);

    ELSIF i=14 THEN              (* SAVE *)

         FillRectangle(h,rmax);
         DosCommand('cls','',okay);
         DosCommand('dir/w','*.fra',okay);
         GetFileName(filename,FRA);
         FillRectangle(h,rmax);
         p.h:=0.0;
         p.v:=0.3;
         WriteString(h,p,"File saved as:");
         p.h:=0.4;
         WriteString(h,p,filename);
         WriteSettings(filename);
         FOR j:=0 TO 60000 DO END;
         FOR j:=0 TO 60000 DO END;

    ELSIF i=15 THEN (* LOAD *)

         FillRectangle(h,rmax);
         DosCommand('cls','',okay);
         DosCommand('dir/w','*.FRA',okay);
         GetFileName(filename,FRA);
         InitFRA(filename);
   END;
   FillRectangle(h,rmax);
  UNTIL menufinished;
END SaveLoadParameters;



PROCEDURE StatusHandler(hojd,bredd:REAL);
VAR r:rectangle;
    p:point;
    b:buttonset;
    k:CARDINAL; (* k only for rubbish *)
    i,j:INTEGER;
```

```
    okay,menufinished:BOOLEAN;
    ch:CHAR;
    string:StringType;
BEGIN
 menufinished:=FALSE;
 FillRectangle(h,rmax);
 REPEAT
   SetTextColor(h,intensewhite);
   p.h:=bredd*31.0;
   p.v:=hojd*23.0*16.0/24.0;
   WriteString(h,p,"STATUS");
   SetTextColor(h,white);
   SetLineColor(h,red);
   p.h:=18.0*bredd;
   p.v:=hojd;
   WriteString(h,p,"BACK TO MAIN MENU");
   p.v:=2.0*hojd;
   WriteString(h,p,"HELP");
   p.v:=14.0*hojd;
   WriteString(h,p,"GENERATOR");
   p.v:=13.0*hojd;
   WriteString(h,p,"ANALYSERS");
   p.v:=12.0*hojd;
   WriteString(h,p,"SWEEP");
   p.v:=11.0*hojd;
   WriteString(h,p,"DISPLAY");
   p.v:=10.0*hojd;
   WriteString(h,p,"PLOT");
   p.v:=9.0*hojd;
   WriteString(h,p,"SYNC");
   p.v:=8.0*hojd;
   WriteString(h,p,"PROGRAM");
   p.v:=7.0*hojd;
   WriteString(h,p,"P");
   r.loleft.h:=11.0*bredd;
   r.upright.h:=16.0*bredd;

   r.loleft.v:=2.0*hojd;
   r.upright.v:=3.0*hojd-0.01;
   DrawRectangle(h,r);
   r.loleft.v:=hojd;
   r.upright.v:=2.0*hojd-0.01;
   DrawRectangle(h,r);
   FOR i:=7 TO 14 DO
     r.loleft.v:=FLOAT(i)*hojd;
     r.upright.v:=FLOAT(i+1)*hojd-0.01;
     DrawRectangle(h,r);
   END;

  ShowCursor;
  i:=WaitMouseRectangle(h);
  r.loleft.v:=FLOAT(i-1)*hojd;
  r.upright.v:=FLOAT(i)*hojd-0.01;
  SetFillColor(h,red);
  HideCursor;
  FillRectangle(h,r);
  SetFillColor(h,black);
  IF i=2 THEN
    menufinished:=TRUE;   (* back to main menu *)

  ELSIF i=3 THEN (* help function *)
     FillRectangle(h,rmax);
     p.h:=0.0;
     p.v:=0.95;
     WriteString(h,p,"HELP STATUS");
     p.h:=0.0;
     p.v:=0.7;
     WriteString(h,p,"Status is your possibility to chech the complete");
     p.h:=0.0;
     p.v:=0.6;
     WriteString(h,p,"FRA status. In 'manual initialisation' you ");
```

```
      p.h:=0.0;
      p.v:=0.5;
      WriteString(h,p,"can see some of the settings as the computer think");
      p.h:=0.0;
      p.v:=0.4;
      WriteString(h,p,"they are. Through STATUS you see all settings as they rea
lly are");
      p.h:=0.7;
      p.v:=0.1;
      WriteString(h,p,"PRESS MOUSE BUTTON TO RETURN");
      WaitForMouse(h,p,b);
    ELSIF (i<16) AND (i>7) THEN
      (* different status enquiries can be sent to the FRA, the answer is
         then  echoed onto the screen until mouse button is pressed    *)
      DosCommand('cls','',okay);
      FillRectangle(h,rmax);
      j:=-i+15;
      CardToString(j,ch,k);
      string[2]:=ch;
      string[0]:='S';
      string[1]:='T';
      show_status:=TRUE;
      SendString(string,3);
      WaitTime(2000);
      WaitForMouse(h,p,b);
      show_status:=FALSE;
   END;
   FillRectangle(h,rmax);
 UNTIL menufinished;
END StatusHandler;



PROCEDURE HMeny(VAR finished:BOOLEAN);
(* huvudmenyn *)
VAR
    r:rectangle;
    b:buttonset;
    p:point;
    bredd,hojd:REAL;
    i,j:CARDINAL;
    filename,oldfile:FileNameType;
    firsttime,okay:BOOLEAN;
    file_status:CHAR;
    s:StringType;
    fil:File;
BEGIN
 firsttime:=TRUE;
 CharacterSize(h,bredd,hojd);
 hojd:=hojd*24.0/16.0;
 REPEAT
   FillRectangle(h,rmax);
   SetTextColor(h,intensewhite);
   p.h:=bredd*35.0;
   p.v:=hojd*23.0*16.0/24.0;
   WriteString(h,p,"MAIN MENU");
   SetTextColor(h,white);
   SetLineColor(h,red);
   p.h:=0.0;
   p.v:=14.0*hojd;
   WriteString(h,p,"INITIALISE:");
   p.v:=9.0*hojd;
   WriteString(h,p,"MEASURE   :");
   p.v:=6.0*hojd;
   WriteString(h,p,"RESULT-    ");
   p.v:=5.0*hojd;
   WriteString(h,p,"OUTPUT    :");
   p.h:=18.0*bredd;
   WriteString(h,p,"BODE DIAGRAM");
   p.v:=6.0*hojd;
   WriteString(h,p,"SAVE/LOAD MEASUREMENT DATA");
```

```
   p.v:=hojd;
   WriteString(h,p,"BACK TO DOS");
   p.v:=2.0*hojd;
   WriteString(h,p,"HELP");
   p.v:=8.0*hojd;
   WriteString(h,p,"ADD DATA FILES");
   p.v:=9.0*hojd;
   WriteString(h,p,"NEW MEASUREMENT");
   p.v:=11.0*hojd;
   WriteString(h,p,"STATUS");
   p.v:=12.0*hojd;
   WriteString(h,p,"SAVE/LOAD PARAMETERS ");
   p.v:=14.0*hojd;
   WriteString(h,p,"MANUAL INITIALISATION");
   p.v:=13.0*hojd;
   WriteString(h,p,"CHAIN");
   p.h:=27.0*bredd;
   IF chain_on THEN
      WriteString(h,p,'IS ON');
   ELSE
      WriteString(h,p,'IS OFF');
   END;
   r.loleft.h:=11.0*bredd;
   r.upright.h:=16.0*bredd;

  FOR i:=2 TO 15 DO
   IF (i<>5) AND (i<>4) THEN
     r.loleft.v:=FLOAT(i-1)*hojd;
     r.upright.v:=FLOAT(i)*hojd-0.01;
     IF firsttime THEN
        SetMouseRectangle(h,r,i);
     END;
     IF (i<>11) AND (i<>8) THEN
        DrawRectangle(h,r);
     END;
   END;
  END;
  firsttime:=FALSE;

  ShowCursor;
  i:=WaitMouseRectangle(h);
  r.loleft.v:=FLOAT(i-1)*hojd;
  r.upright.v:=FLOAT(i)*hojd-0.01;
  SetFillColor(h,red);
  HideCursor;
  FillRectangle(h,r);
  SetFillColor(h,black);
  CASE i OF
   2:
        finished:=TRUE;
|
   3:   FillRectangle(h,rmax);
        p.h:=0.0;
        p.v:=0.95;
        WriteString(h,p,"HELP MAIN MENU");
        p.v:=0.8;
        WriteString(h,p,"* MANUAL INITIALISATION: check and set important param
eters.");
        p.v:=0.75;
        WriteString(h,p,"* CHAIN: Connect parameter files saved on disc.");
        p.v:=0.7;
        WriteString(h,p,"         This makes multiple measurements possible.");
        p.v:=0.65;
        WriteString(h,p,"* SAVE/LOAD PARAMETERS: Present settings are saved ont
o a file");
        p.v:=0.6;
        WriteString(h,p,"                        in your default library.");
        p.v:=0.55;
        WriteString(h,p,"* STATUS: Enables you to check all parameters.");
        p.v:=0.5;
        WriteString(h,p,"* NEW MEASUREMENT: Makes measurement. Result is writte
```

```
n to TEMP.MEA.");
        p.v:=0.45;
        WriteString(h,p,"* ADD DATA FILES: Data in TEMP.MEA is sorted into");
        p.v:=0.4;
        WriteString(h,p,"                  an old file You specify.");
        p.v:=0.35;
        WriteString(h,p,"");
        p.v:=0.35;
        WriteString(h,p,"* SAVE/LOAD DATA: to or from TEMP.MEA.");
        p.v:=0.3;
        WriteString(h,p,"* BODE DIAGRAM: Bode plot made whith data stored in TE
MP.MEA.");
        p.v:=0.25;
        WriteString(h,p,"* HELP: This text.");
        p.v:=0.2;
        WriteString(h,p,"* BACK TO DOS: Leave this program. Parameters may be l
ost but not data.");
        p.h:=0.7;
        p.v:=0.05;
        WriteString(h,p,"PRESS MOUSE BUTTON TO RETURN");
        WaitForMouse(h,p,b);
|
   6:    BodeDiagram;
|
   7:    SaveLoadData(hojd,bredd);

|
   8:|    (* nop *)
   9:  (* measurement to add... *)
        FillRectangle(h,rmax);
        DosCommand('cls','',okay);
        DosCommand('dir/w','*.mea',okay);
        p.h:=0.0;
        p.v:=0.55;
        WriteString(h,p,'To what existing file.MEA shall TEMP.MEA be added?');
        GetFileName(filename,MEA);
        oldfile:=filename;
        oldfile[12]:='E';
        Lookup(fil,filename,FALSE);
        Rename(fil,oldfile);
        Close(fil);
        p.h:=0.0;
        p.v:=0.1;
        WriteString(h,p,'Sorting files together. ');
        SortTogether(filename,oldfile);

|
   10: IF chain_on THEN
         ChainMeasure;
       ELSE
         j:=Pos('MA',settings);
         ExtractValue(s,j);
         StringToReal(s,EndFreq,okay);
         Measure;
       END;
|
   11:| (* har skall den inte gora nagot alls *)

   12:  (* status direct from the FRA *)
        StatusHandler(hojd,bredd);
|
   13:  SaveLoadParameters(hojd,bredd);
|
   14:  ChainMenu(bredd,hojd);
|
   15:  ManualInitMenu(bredd,hojd);
  END;
 UNTIL finished;

END HMeny;
```

```
PROCEDURE ChainMeasure;
(* utfoer en hel serie maetningar *)

VAR fil,summafil,chainfil,delfil:File;
    filnamn,maa,mee:FileNameType;
    i,j:CARDINAL;
    ch:CHAR;
    done,okay:BOOLEAN;
    s:StringType;
BEGIN
(* I TEMP.MEA hamnar delresultat *)
(* Alla delresultat samlas upp i temp.maa. Denna faar efter sista maetningen
   byta namn till temp.mea som aer procedurens resultatfil *)
(* Efter varje delmaetning (utom den foersta) sorteras temp.mea och temp.maa
   in i temp.mee (tillfaellig sorteringsfil). Temp.mee faar sedan byta namn
   till temp.maa som ju var uppsamlaren. *)
     s:='TEMP    .MEE';
     FOR i:=1 TO 12 DO
       maa[i]:=s[i-1];
       mee[i]:=s[i-1];
     END;
     maa[11]:='a';
     maa[12]:='a';
     done:=FALSE;
     FillRectangle(h,rmax);
     p.h:=0.5;
     p.v:=0.5;
     WriteString(h,p,'Multiple measurement. CHAIN ON');

     DosCommand('del','temp.mea',okay);
     IF NOT okay THEN
       FillRectangle(h,rmax);
       p.h:=0.5;
       p.v:=0.2;
       WriteString(h,p,'DosCommand(1) ERROR ChainMeasure');
     END;
     (* se till att temp.mee existerar *)
     Lookup(fil,'temp.mee',TRUE);
     Close(fil);

     Lookup(chainfil,'temp.chn',TRUE);
     (* foersta maetningen *)
     FOR i:=1 TO 12 DO
       REPEAT
         ReadChar(chainfil,ch);
         IF ch=CHR(0) THEN
           done:=TRUE;
         END;
       UNTIL done OR ((ch<='z') AND (ch>=' '));
       filnamn[i]:=ch;
     END;
     ReadChar(chainfil,ch); (* EOL *)
     IF NOT done THEN
       InitFRA(filnamn);
       i:=Pos('MA',settings);
       ExtractValue(s,i);
       StringToReal(s,EndFreq,okay);
       IF NOT okay THEN
         FillRectangle(h,rmax);
         p.h:=0.5;
         p.v:=0.2;
         WriteString(h,p,'StringToReal(1) ERROR ChainMeasure');
         WaitTime(1000);
       END;
       Measure;
       Lookup(delfil,'TEMP.MEA',FALSE);
       Rename(delfil,'TEMP.MAA');
       Close(delfil);
     END;
     (* resten av maetningarna *)
```

```
    WHILE NOT done DO
       FOR i:=1 TO 12 DO
         REPEAT
           ReadChar(chainfil,ch);
           IF ch=CHR(0) THEN
             done:=TRUE;
           END;
         UNTIL done OR ((ch<='z') AND (ch>=' '));
         filnamn[i]:=ch;
       END;
       ReadChar(chainfil,ch); (* EOL *)
       IF NOT done THEN
          InitFRA(filnamn);
          i:=Pos('MA',settings);
          ExtractValue(s,i);
          StringToReal(s,EndFreq,okay);
          IF NOT okay THEN
            FillRectangle(h,rmax);
            p.h:=0.5;
            p.v:=0.2;
            WriteString(h,p,'StringToReal(2) ERROR ChainMeasure');
          END;
          Measure;
          SortTogether(mee,maa);
          DosCommand('del','temp.maa',okay);
          IF NOT okay THEN
            FillRectangle(h,rmax);
            p.h:=0.5;
            p.v:=0.2;
            WriteString(h,p,'DosComand(2) ERROR ChainMeasure');
          END;
          Lookup(summafil,'TEMP.MEE',FALSE);
          Rename(summafil,'TEMP.MAA');
          Close(summafil);
          Lookup(fil,'TEMP.MEE',TRUE);  (* aaterskapa tom fil *)
          Close(fil);
       END;
    END;
    DosCommand('del','temp.mea',okay);
    Lookup(summafil,'TEMP.MAA',FALSE);
    Rename(summafil,'TEMP.MEA');
    Close(summafil);
    Close(chainfil);
END ChainMeasure;

PROCEDURE SortTogether(into,from:FileNameType);
(* The file TEMP.MEA always contains the last measurement made.  *)
(* 'from' and TEMP.MEA are sorted in frequency order into 'into' *)
VAR new_data,old_data,dest:File;
    j,i:CARDINAL;
    new_str,old_str:StringType;
    new_val,old_val:REAL;
    okay,old_end,new_end:BOOLEAN;
    ch:CHAR;
BEGIN
  Lookup(new_data,'TEMP.MEA',FALSE); (* TEMP.MEA contains data from a *)
                                     (* measurement just taken.       *)
  Lookup(old_data,from,FALSE);       (* 'from' is an old data file.   *)
  Lookup(dest,into,TRUE);            (* Result file!                  *)

  old_end:=FALSE;       (* old_end=TRUE means no more data in old_data *)
  new_end:=FALSE;       (* new_end=TRUE means no more data in new_data *)
  (* search for first frequency value in each file *)
  REPEAT
    ReadChar(old_data,ch);
  UNTIL (ch='+') OR (ch=CHR(0));
  IF ch=CHR(0) THEN
     old_end:=TRUE;
  ELSE
    old_str[0]:=ch;
    i:=1;
```

```
  WHILE i<11 DO
    IF NOT old_end THEN
      ReadChar(old_data,old_str[i]);
      IF old_str[i]=CHR(0) THEN
        old_end:=TRUE;
      END;
    END;
    INC(i);
  END;
  old_str[11]:=CHR(0);
END;

(* the same again but the other file *)
REPEAT
  ReadChar(new_data,ch);
UNTIL (ch='+') OR (ch=CHR(0));
IF ch=CHR(0) THEN
   new_end:=TRUE;
ELSE
  new_str[0]:=ch;
  i:=1;
  WHILE i<11 DO
    IF NOT new_end THEN
      ReadChar(new_data,new_str[i]);
      IF new_str[i]=CHR(0) THEN
         new_end:=TRUE;
      END;
    END;
    INC(i);
  END;
  new_str[11]:=CHR(0);
END;
(* if the first frequencies exist then convert them to reals *)
IF NOT new_end THEN
  StringToReal(new_str,new_val,okay);
END;
IF NOT old_end THEN
  StringToReal(old_str,old_val,okay);
END;

(* sorting while both files still have data *)
WHILE (NOT new_end) AND (NOT old_end) DO
  IF new_val<old_val THEN
    FOR i:=0 TO 10 DO
      WriteChar(dest,new_str[i]);
    END;
    FOR i:=1 TO 26 DO
      ReadChar(new_data,ch);
      WriteChar(dest,ch);
    END;
    WriteChar(dest,EOL);
    REPEAT
      ReadChar(new_data,ch);
    UNTIL (ch='+') OR (ch='-') OR (ch=CHR(0));
    IF ch=CHR(0) THEN
      new_end:=TRUE;
    ELSE
      new_str[0]:=ch;
      FOR i:=1 TO 10 DO
        ReadChar(new_data,new_str[i]);
      END;
      new_str[11]:=CHR(0);
      StringToReal(new_str,new_val,okay);
    END;
  ELSE
    FOR i:=0 TO 10 DO
      WriteChar(dest,old_str[i]);
    END;
    FOR i:=1 TO 26 DO
        ReadChar(old_data,ch);
        WriteChar(dest,ch);
```

```
      END;
      WriteChar(dest,EOL);
      REPEAT
        ReadChar(old_data,ch);
      UNTIL (ch='+') OR (ch='-') OR (ch=CHR(0));
      IF ch=CHR(0) THEN
         old_end:=TRUE;
      ELSE
        old_str[0]:=ch;
        FOR i:=1 TO 10 DO
          ReadChar(old_data,old_str[i]);
        END;
        old_str[11]:=CHR(0);
        StringToReal(old_str,old_val,okay);
      END;
    END;
  END;
  (* now only one of the files has data left *)
  IF old_end THEN
   FOR i:=0 TO 10 DO
     WriteChar(dest,new_str[i]);
   END;
   REPEAT
    ReadChar(new_data,ch);
    WriteChar(dest,ch);
   UNTIL ch=CHR(0);
  ELSE
   FOR i:=0 TO 10 DO
     WriteChar(dest,old_str[i]);
   END;
   REPEAT
    ReadChar(old_data,ch);
    WriteChar(dest,ch);
   UNTIL ch=CHR(0);
  END;
  Close(new_data);
  Close(old_data);
  Close(dest);
  Delete(from,old_data);

END SortTogether;


(* PROCEDURES beneath down to BODEDIAGRAM belong to BodeDiagram *)

  PROCEDURE HorizontalFreqPos(maxfreq,minfreq:REAL;VAR workfreq:REAL);
  (* BODE:workfreq is both input and output variable *)
  BEGIN
    workfreq:=ln(workfreq)/ln(10.0);
    workfreq:=1.4*(workfreq-minfreq)/(maxfreq-minfreq)+0.1;
  END HorizontalFreqPos;

  PROCEDURE VerticalAmpPos(max,min:REAL;VAR workr:REAL);
  (* BODE:workr is both input and output variable *)
  BEGIN
    workr:=ln(workr)/ln(10.0);
    workr:=0.9*(workr-min)/(max-min)+0.1;
  END VerticalAmpPos;

  PROCEDURE VerticalPhasePos(max,min:REAL;VAR work:REAL);
  (* BODE:work is both input and output variable *)
  BEGIN
    work:=0.9*(work-min)/(max-min)+0.1;
  END VerticalPhasePos;

  PROCEDURE CutEven(VAR maxr,minr,maxtheta,mintheta,maxfreq,minfreq:REAL);
  (* BODE: Puts variables into even decades etc *)
  VAR minus:BOOLEAN;
  BEGIN
    minus:=FALSE;
    (* logaritmering *)
```

```
  maxr:=ln(maxr)/ln(10.0);
  minr:=ln(minr)/ln(10.0);
  maxfreq:=ln(maxfreq)/ln(10.0);
  minfreq:=ln(minfreq)/ln(10.0);

  (* fasen skall vara jaemna moduler av 90 grader *)
  IF maxtheta>0.0 THEN
    maxtheta:=90.0*(FLOAT(TRUNC(maxtheta/90.0))+1.0);
  ELSE
    maxtheta:=-90.0*(FLOAT(TRUNC(ABS(maxtheta/90.0))));
  END;
  IF mintheta<0.0 THEN
    mintheta:=-90.0*(FLOAT(TRUNC(ABS(mintheta/90.0)))+1.0);
  ELSE
    mintheta:=90.0*FLOAT(TRUNC(mintheta/90.0));
  END;

  (* ampl skall vara jaemna moduler av 10 decilog *)

  IF maxr<0.0 THEN
     maxr:=-FLOAT(TRUNC(ABS(maxr)));
  ELSE
    maxr:=FLOAT(TRUNC(maxr))+1.0;
  END;

  IF minr<0.0 THEN
     minr:=-FLOAT(TRUNC(ABS(minr)))-1.0;
  ELSE
    minr:=FLOAT(TRUNC(minr));
  END;


  (* frekvensen skall vara jaemna dekader *)
  IF maxfreq<0.0 THEN
     maxfreq:=-FLOAT(TRUNC(ABS(maxfreq)));
  ELSIF maxfreq<>FLOAT(TRUNC(maxfreq)) THEN
    maxfreq:=FLOAT(TRUNC(maxfreq))+1.0;
  END;

  IF minfreq<0.0 THEN
     minfreq:=-FLOAT(TRUNC(ABS(minfreq)))-1.0;
  ELSE
    minfreq:=FLOAT(TRUNC(minfreq));
  END;

END CutEven;


PROCEDURE EndTest(ch:CHAR; VAR endstate:CARDINAL);
(* BODE: sets endstate to endstate+1 or to zero  *)
BEGIN
  IF (ORD(ch)=0) OR (ORD(ch)=30) OR (ORD(ch)=32) OR (ORD(ch)=36) THEN
     INC(endstate);
  ELSE
     endstate:=0;
  END;
END EndTest;

PROCEDURE OkayTest(okay:BOOLEAN;radnr,workstring:ARRAY OF CHAR);
(* radnr=foer felutskrift *)
VAR  b:buttonset;
BEGIN
 IF NOT okay THEN
    p.v:=0.5;
    p.h:=0.5;
    WriteString(h,p,'not okay line ');
    p.v:=0.5;
    p.h:=0.78;
    WriteString(h,p,radnr);
    p.v:=0.7;
    p.h:=0.58;
```

```
      WriteString(h,p,workstring);
      WaitForMouse(h,p,b);  (* press mouse button to return *)
      FillRectangle(h,rmax);
    END;
  END OkayTest;


PROCEDURE BodeDiagram;
VAR maxr,minr,maxtheta,mintheta,maxfreq,minfreq:REAL;
    workfreq,workr,worktheta:REAL;
    workstring:StringType;
    b:buttonset;
    okay:BOOLEAN;
    p:point;
    fil:File;
    endstate:CARDINAL; (* The value of endstate tells how many {0,30,32,36}
                          have past since last useful sign. EOF is definetly
                          reached when the value is 3. This solution is
                          chosen so that manual editing of the file is
                          possible. If endstate>0 then it means that we
                          are not busy reading a value *)
    laststate:CARDINAL; (* the endstate before the first +/- in a real.
                          If we read a useful sign and laststate>0 then
                          it means than it is the first sign in a new real *)
    valuemeaning:CARDINAL; (* state variable that keeps track of what
                              column in the file is to be read.
                              Possible values {0,1,2} *)
    ch:CHAR;
    line:ARRAY [0..1] OF point;
    dot :ARRAY [0..0] OF point;
    i:CARDINAL;


  PROCEDURE FileMaxMin(VAR max,min:REAL;VAR fil:File;column:CARDINAL);
  (* finds max and min values in the file "fil". Column number starts
     with 0 as the first column to search *)

  VAR workvalue:REAL;
      i:CARDINAL;
  BEGIN
    endstate:=1;
    min:=1.0E30;
    max:=-1.0E30;
    valuemeaning:=0;
    Reset(fil);
    REPEAT
      ReadChar(fil,ch);
      laststate:=endstate;
      EndTest(ch,endstate);
      IF ((ch="+") OR (ch="-")) AND (endstate=0) AND (laststate<>0) THEN
        IF valuemeaning=column THEN
           i:=1;
           workstring[0]:=ch;
           REPEAT
             ReadChar(fil,ch);
             EndTest(ch,endstate);
             IF endstate=0 THEN
                workstring[i]:=ch;
                INC(i);
             END;
           UNTIL endstate>0;
           WHILE i<=11 DO
              workstring[i]:=CHR(0);
              INC(i);
           END;
           StringToReal(workstring,workvalue,okay);
           OkayTest(okay,'183',workstring);
           IF workvalue<min THEN min:=workvalue; END;
           IF workvalue>max THEN max:=workvalue; END;
        END;
        INC(valuemeaning);
```

```
      valuemeaning:=valuemeaning MOD 3;
    END;
  UNTIL endstate=3;
END FileMaxMin;



PROCEDURE Scale;
(* subrutin som ritar skalan till bodediagrammet *)
BEGIN
  FileMaxMin(maxfreq,minfreq,fil,0);
  FileMaxMin(maxr,minr,fil,1);
  FileMaxMin(maxtheta,mintheta,fil,2);
  CutEven(maxr,minr,maxtheta,mintheta,maxfreq,minfreq);
  RealToString(minfreq,0,12,workstring,okay);
  OkayTest(okay,'203',workstring);
  p.v:=0.0;
  p.h:=-0.08;
  WriteString(h,p,workstring);
  RealToString(maxfreq,0,12,workstring,okay);
  OkayTest(okay,'199',workstring);
  p.h:=1.25;
  WriteString(h,p,workstring);
  SetTextColor(h,lightred);
  RealToString(minr*10.0,0,12,workstring,okay);
  OkayTest(okay,'204',workstring);
  p.v:=0.1;
  p.h:=-0.135;
  WriteString(h,p,workstring);
  RealToString(maxr*10.0,0,12,workstring,okay);
  OkayTest(okay,'217',workstring);
  p.v:=0.965;
  WriteString(h,p,workstring);
  SetTextColor(h,white);

  (* Drawing the vertical axis for the amplidude. *)
  SetLineColor(h,lightred);
  line[0].h:=0.1;
  line[0].v:=0.1;
  line[1].h:=0.1;
  line[1].v:=1.0;
  PolyLine(h,line,2);

  workr:=minr;
  WHILE workr<maxr DO
    worktheta:=workr;
    workr:=0.9*(workr-minr)/(maxr-minr)+0.1;
    line[0].h:=0.1;
    line[1].h:=0.12;
    line[0].v:=workr;
    line[1].v:=workr;
    PolyLine(h,line,2);
    workr:=worktheta+0.3;
    workr:=0.9*(workr-minr)/(maxr-minr)+0.1;
    line[1].h:=0.11;
    line[0].v:=workr;
    line[1].v:=workr;
    PolyLine(h,line,2);
    workr:=worktheta+0.7;
    workr:=0.9*(workr-minr)/(maxr-minr)+0.1;
    line[0].v:=workr;
    line[1].v:=workr;
    PolyLine(h,line,2);
    workr:=worktheta+1.0;
  END; (* ampl axis drawn *)

  (* drawing the horizontal axis and scale = frequency *)
  SetLineColor(h,white);
  line[0].h:=0.1;
  line[0].v:=0.1;
  line[1].h:=1.5;
```

```
    line[1].v:=0.1;
    PolyLine(h,line,2);

    workfreq:=minfreq;
    WHILE workfreq<maxfreq DO
      worktheta:=workfreq;
      workfreq:=1.4*(workfreq-minfreq)/(maxfreq-minfreq)+0.1;
      line[0].v:=0.1;
      line[1].v:=0.12;
      line[0].h:=workfreq;
      line[1].h:=workfreq;
      PolyLine(h,line,2);
      workfreq:=worktheta+0.3;
      workfreq:=1.4*(workfreq-minfreq)/(maxfreq-minfreq)+0.1;
      line[1].v:=0.11;
      line[0].h:=workfreq;
      line[1].h:=workfreq;
      PolyLine(h,line,2);
      workfreq:=worktheta+0.7;
      workfreq:=1.4*(workfreq-minfreq)/(maxfreq-minfreq)+0.1;
      line[0].h:=workfreq;
      line[1].h:=workfreq;
      PolyLine(h,line,2);
      workfreq:=worktheta+1.0;
    END;        (* freq axis drawn *)

    (* Drawing the vertical axis and scale for the phase. *)
    SetLineColor(h,brown);
    line[0].h:=1.5;
    line[0].v:=0.1;
    line[1].h:=1.5;
    line[1].v:=1.0;
    PolyLine(h,line,2);
    worktheta:=mintheta+90.0;
    WHILE worktheta<maxtheta DO
      workr:=worktheta;
      VerticalPhasePos(maxtheta,mintheta,worktheta);
      line[0].h:=1.48;
      line[1].h:=1.5;
      line[0].v:=worktheta;
      line[1].v:=worktheta;
      PolyLine(h,line,2);
      worktheta:=workr+90.0;
    END;
    SetLineColor(h,red);
    SetTextColor(h,brown);
    RealToString(mintheta,0,12,workstring,okay);
    OkayTest(okay,'330',workstring);
    p.v:=0.05;
    p.h:=1.28;
    WriteString(h,p,workstring);
    RealToString(maxtheta,0,12,workstring,okay);
    OkayTest(okay,'335',workstring);
    p.v:=0.965;
    p.h:=1.28;
    WriteString(h,p,workstring);
    SetTextColor(h,white);
    (* phase axis drawn *)
  END Scale;

BEGIN   (* BodeDiagram *)
  FillRectangle(h,rmax);
  Lookup(fil,"temp.mea",FALSE); (* temp.mea is the workfile containing data *)
  Scale;
  (* the bode plot itself *)
  (* first the amplitude plot *)
    Reset(fil);
    endstate:=1;
    valuemeaning:=0;
    REPEAT
      ReadChar(fil,ch);
```

```
    FOR i:=0 TO 11 DO
       workstring[i]:=CHR(0);
    END;
    laststate:=endstate;
    EndTest(ch,endstate);
    IF ((ch="+") OR (ch="-")) AND (endstate=0) AND (laststate<>0) THEN
      IF valuemeaning=0 THEN
         i:=1;
         workstring[0]:=ch;
         REPEAT
           ReadChar(fil,ch);
           EndTest(ch,endstate);
           IF endstate=0 THEN
              workstring[i]:=ch;
              INC(i);
           END;
         UNTIL endstate>0;
         StringToReal(workstring,workfreq,okay);
         OkayTest(okay,'372',workstring);
      ELSIF   valuemeaning=1 THEN
         i:=1;
         workstring[0]:=ch;
         REPEAT
           ReadChar(fil,ch);
           EndTest(ch,endstate);
           IF endstate=0 THEN
              workstring[i]:=ch;
              INC(i);
           END;
         UNTIL endstate>0;
         StringToReal(workstring,workr,okay);
         OkayTest(okay,'385',workstring);

         (* as r comes after freq in a row, we now have a dot to plot *)
         HorizontalFreqPos(maxfreq,minfreq,workfreq);
         VerticalAmpPos(maxr,minr,workr);
         dot[0].v:=workr;
         dot[0].h:=workfreq;
         PolyMarker(h,dot,1);
      END;
      INC(valuemeaning);
      valuemeaning:=valuemeaning MOD 3;
    END;
  UNTIL endstate=3;   (* EOF reached and bode (gain) plot made *)
(* then the phase plot *)
  SetLineColor(h,yellow);
  Reset(fil);
  endstate:=1;
  valuemeaning:=0;
  REPEAT
    ReadChar(fil,ch);
    FOR i:=0 TO 11 DO
       workstring[i]:=CHR(0);
    END;
    laststate:=endstate;
    EndTest(ch,endstate);
    IF ((ch="+") OR (ch="-")) AND (endstate=0) AND (laststate<>0) THEN
      IF valuemeaning=0 THEN
         i:=1;
         workstring[0]:=ch;
         REPEAT
           ReadChar(fil,ch);
           EndTest(ch,endstate);
           IF endstate=0 THEN
              workstring[i]:=ch;
              INC(i);
           END;
         UNTIL endstate>0;
         StringToReal(workstring,workfreq,okay);
         OkayTest(okay,'423',workstring);
      ELSIF   valuemeaning=2 THEN
```

```
            i:=1;
            workstring[0]:=ch;
            REPEAT
              ReadChar(fil,ch);
              EndTest(ch,endstate);
              IF endstate=0 THEN
                 workstring[i]:=ch;
                 INC(i);
              END;
            UNTIL endstate>0;
            StringToReal(workstring,worktheta,okay);
            OkayTest(okay,'436',workstring);
            (* as theta comes after freq in a row, we now have a dot to plot *)
            HorizontalFreqPos(maxfreq,minfreq,workfreq);
            VerticalPhasePos(maxtheta,mintheta,worktheta);
            dot[0].v:=worktheta;
            dot[0].h:=workfreq;
            PolyMarker(h,dot,1);
          END;
          INC(valuemeaning);
          valuemeaning:=valuemeaning MOD 3;
        END;
      UNTIL endstate=3;   (* EOF reached and bode (phase) plot made *)
      SetLineColor(h,red);
      WaitForMouse(h,p,b);  (* press mouse button to return *)

END BodeDiagram;



PROCEDURE GetValue(VAR value:StringType;
                   lo,hi:StringType;numbertype:NumberType);
(* inmatningsrutin for cardinal eller real *)
(* lo och hi aer minimalt och maximalt tillaatna vaerden *)
VAR p:point;                    (* cursorns lage *)
    workstring:StringType;      (* innehaller den ofiltrerade strangen *)
    nrofchar:INTEGER;           (* raknar antal filtrerade/godkanda tecken *)
    i:INTEGER;                  (* vanlig raknevariabel *)
    r:rectangle;
    okay:BOOLEAN;
    locard,hicard,wcard:CARDINAL;
    loreal,hireal,wreal:REAL;
BEGIN
  r.loleft.v:=0.2;
  r.loleft.h:=1.025;
  r.upright.v:=0.6;
  r.upright.h:=1.6;
  IF numbertype=cardinal THEN
     StringToCard(lo,locard,okay);
     StringToCard(hi,hicard,okay);
  ELSE
     StringToReal(lo,loreal,okay);
     StringToReal(hi,hireal,okay);
  END;
  REPEAT
     FOR i:=0 TO 11 DO
       value[i]:=" ";
     END;
     i:=-1;
     nrofchar:=-1;
     DrawRectangle(h,r);
     p.h:=1.05;
     p.v:=0.45;
     WriteString(h,p,"NEW VALUE:");
     p.v:=0.4;
     ReadString(h,p,workstring);
     REPEAT  (* workstring filtreras och godkanda tecken kopieras till value*)
       INC(i);
       IF numbertype=cardinal THEN
         IF ((workstring[i]>="0") AND (workstring[i]<="9"))  THEN
                    INC(nrofchar);
```

```
                    value[nrofchar]:=workstring[i];
          END;
        ELSE      (* numbertype=real *)
          IF ((workstring[i]>="0") AND (workstring[i]<="9")) OR
             (workstring[i]="E") OR (workstring[i]="-") OR
             (workstring[i]=".") OR (workstring[i]="+") THEN
                    INC(nrofchar);
                    value[nrofchar]:=workstring[i];
          END;
        END;

      UNTIL (i=11) OR (ORD(workstring[i])=0);
      FillRectangle(h,r);
      IF numbertype=cardinal THEN
        StringToCard(workstring,wcard,okay);
        IF NOT (okay AND (wcard>=locard) AND (wcard<=hicard)) THEN
          okay:=FALSE;
          p.v:=0.35;
          WriteString(h,p,"OUT OF RANGE !");
          p.v:=0.3;
          WriteString(h,p,lo);
          p.h:=1.17;
          WriteString(h,p,"<x<");
          p.h:=1.24;
          WriteString(h,p,hi);
        END;
      ELSE
         StringToReal(workstring,wreal,okay);
         IF NOT (okay AND (wreal>=loreal) AND (wreal<=hireal)) THEN
           okay:=FALSE;
           p.v:=0.35;
           WriteString(h,p,"OUT OF RANGE !");
           p.v:=0.3;
           WriteString(h,p,lo);
           p.h:=1.17;
           WriteString(h,p,"<x<");
           p.h:=1.24;
           WriteString(h,p,hi);
         END;
       END;
  UNTIL okay;

END GetValue;


PROCEDURE RemoveSettingIfExisting(inputstring:StringType);
(* tar bort ett kommando ur settings-strangen *)
(* Avsedd for nytt kommando som inte har samma bokstaver som det gamla. *)
VAR command:ARRAY [0..1] OF CHAR;
    j,position,oldparameterlength:CARDINAL;

BEGIN
   oldparameterlength:=0;
   command[0]:=inputstring[0];
   command[1]:=inputstring[1];
   position:=Pos(command,settings);
   IF position<HIGH(settings) THEN  (* om command finns i settings *)
      j:=position;
      REPEAT    (* tar reda pa gamla kommandots langd *)
         INC(oldparameterlength);
         INC(j);
      UNTIL settings[j-1]=";";
      (* skriv over gamla kommandot sa att inga blanktecken blir kvar *)
      FOR j:=0 TO settingcounter-position DO
          settings[position+j]:=settings[position+j+oldparameterlength];
      END;
      (* settings-strangen ar kortare nu ju *)
      settingcounter:=settingcounter-oldparameterlength;
      (* fyll ut settings med blanka pa slutet *)
      FOR j:=settingcounter TO 200 DO
          settings[j]:=" ";
```

```
      END;
    END;
END RemoveSettingIfExisting;


PROCEDURE AmplMenu(bredd,hojd:REAL);
VAR r:rectangle;
    p:point;
    b:buttonset;
    k:CARDINAL; (* k only for rubbish *)
    i,j:INTEGER;
    okay,menufinished:BOOLEAN;
    ch:CHAR;
    string:StringType;
    value:StringType;
BEGIN
 menufinished:=FALSE;
 FillRectangle(h,rmax);
 REPEAT
   SetTextColor(h,intensewhite);
   p.h:=bredd*31.0;
   p.v:=hojd*23.0*16.0/24.0;
   WriteString(h,p,"AMPLITUDE");
   SetTextColor(h,white);
   SetLineColor(h,red);
   p.h:=18.0*bredd;
   p.v:=hojd;
   WriteString(h,p,"BACK TO PREVIOUS MENU");
   p.v:=2.0*hojd;
   WriteString(h,p,"HELP");
   p.v:=14.0*hojd;
   WriteString(h,p,"AMPLITUDE");
   p.h:=0.0;
   WriteString(h,p,"COMPR OFF:");
   p.v:=11.0*hojd;
   WriteString(h,p,"COMPR ON :");
   p.h:=18.0*bredd;
   p.v:=12.0*hojd;
   WriteString(h,p,"AMPLITUDE COMPRESSION ON/OFF");
   p.v:=11.0*hojd;
   WriteString(h,p,"SOURCE");
   p.v:=10.0*hojd;
   WriteString(h,p,"VALUE");
   p.v:=9.0*hojd;
   WriteString(h,p,"ERROR");
   r.loleft.h:=11.0*bredd;
   r.upright.h:=16.0*bredd;
   AmplValues(bredd,hojd);
   r.loleft.v:=14.0*hojd;
   r.upright.v:=15.0*hojd-0.01;
   DrawRectangle(h,r);
   r.loleft.v:=2.0*hojd;
   r.upright.v:=3.0*hojd-0.01;
   DrawRectangle(h,r);
   r.loleft.v:=hojd;
   r.upright.v:=2.0*hojd-0.01;
   DrawRectangle(h,r);
   FOR i:=9 TO 12 DO
     r.loleft.v:=FLOAT(i)*hojd;
     r.upright.v:=FLOAT(i+1)*hojd-0.01;
     DrawRectangle(h,r);
   END;

  ShowCursor;
  i:=WaitMouseRectangle(h);
  r.loleft.v:=FLOAT(i-1)*hojd;
  r.upright.v:=FLOAT(i)*hojd-0.01;
  SetFillColor(h,red);
  HideCursor;
  FillRectangle(h,r);
  SetFillColor(h,black);
```

```
r.loleft.h:=1.025;    (* koordinater for inforuta *)
r.loleft.v:=0.5;
r.upright.h:=1.5;
r.upright.v:=0.75;
IF i=2 THEN
  menufinished:=TRUE;   (* back to main menu *)

ELSIF i=3 THEN (* help function *)
   FillRectangle(h,rmax);
   p.h:=0.0;
   p.v:=0.95;
   WriteString(h,p,"HELP AMPLITUDE");
   p.h:=0.0;
   p.v:=0.7;
   WriteString(h,p,"* AMPLITUDE COMPRESSION OFF: only AMPLITUDE matters.");
   p.h:=0.0;
   p.v:=0.6;
   WriteString(h,p,"* AMPLITUDE COMPRESSION ON : Generator tries to hold ");
   p.h:=0.0;
   p.v:=0.5;
   WriteString(h,p,"  the amplitude VALUE within  a %-ERROR whith channel");
   p.h:=0.0;
   p.v:=0.4;
   WriteString(h,p,"  SOURCE as source.");
   p.h:=0.7;
   p.v:=0.1;
   WriteString(h,p,"PRESS MOUSE BUTTON TO RETURN");
   WaitForMouse(h,p,b);
 ELSIF i= 15 THEN   (* amplitud *)
       DrawRectangle(h,r);
       p.h:=1.05;
       p.v:=0.6;
       WriteString(h,p,"SETTING OF AMPLITUDE");
       GetValue(value,'0.0','10.23',real);
       Insert("AM",string,0);
       Insert(value,string,2);
       SetFRA(string);
ELSIF i=13 THEN (* ampl compr on/off *)
       DrawRectangle(h,r);
       p.h:=1.05;
       p.v:=0.6;
       WriteString(h,p,"COMPRESSION");
       p.h:=1.05;
       p.v:=0.55;
       WriteString(h,p,"0=OFF");
       p.h:=1.05;
       p.v:=0.5;
       WriteString(h,p,"1=ON");
       GetValue(value,'0','1',cardinal);
       Insert("AP",string,0);
       Insert(value,string,2);
       SetFRA(string);
ELSIF i=12 THEN  (* source *)
       DrawRectangle(h,r);
       p.h:=1.05;
       p.v:=0.6;
       WriteString(h,p,"SOURCE");
       p.h:=1.05;
       p.v:=0.55;
       WriteString(h,p,"1=channel 1");
       p.h:=1.05;
       p.v:=0.5;
       WriteString(h,p,"2=channel 2");
       GetValue(value,'1','2',cardinal);
       Insert("AS",string,0);
       Insert(value,string,2);
       Insert("00",string,3);
       SetFRA(string);
ELSIF i=11 THEN  (* value *)
       DrawRectangle(h,r);
       p.h:=1.05;
```

```
        p.v:=0.6;
        WriteString(h,p,"SETTING OF VALUE");
        GetValue(value,'1E-4','300.0',real);
        Insert("AV",string,0);
        Insert(value,string,2);
        SetFRA(string);
  ELSIF i=10 THEN  (* error *)
        DrawRectangle(h,r);
        p.h:=1.05;
        p.v:=0.6;
        WriteString(h,p,"SETTING OF ERROR");
        p.h:=1.05;
        p.v:=0.5;
        WriteString(h,p,"VALUE IN %");
        GetValue(value,'1.0','50.0',real);
        Insert("AE",string,0);
        Insert(value,string,2);
        SetFRA(string);
  END;
  FillRectangle(h,rmax);
 UNTIL menufinished;

END AmplMenu;



PROCEDURE ManualInitMenu(bredd,hojd:REAL);
(* meny och rutiner for att satta enskilda parametrar i frekvensanalysatorn *)

VAR r:rectangle;
    b:buttonset;
    p:point;
    i,j:CARDINAL;
    menufinished,firsttime:BOOLEAN;
    parameter:StringType;
    string,value:StringType;


 PROCEDURE IntegrationSetting(r:rectangle);
 VAR p:point;
 BEGIN
        DrawRectangle(h,r);
        p.h:=1.05;
        p.v:=0.7;
        WriteString(h,p,"INTEGRATION TIME");
        p.v:=0.65;
        WriteString(h,p,"  SECONDS  =0");
        p.v:=0.6;
        WriteString(h,p,"  CYCLES   =1");
        p.v:=0.55;
        WriteString(h,p,"  AUTOMATIC=2");
        GetValue(value,'0','2',cardinal);
        IF value[0]="0" THEN
           RemoveSettingIfExisting("IC");
          (* inmatningrutin for ny fordrojning *)
           p.v:=0.7;
           WriteString(h,p,"INTEGRATION TIME:");
           p.v:=0.65;
           EraseChar(h,p,15);
           WriteString(h,p,"VALUE IN SECONDS");
           p.v:=0.6;
           EraseChar(h,p,15);
           p.v:=0.55;
           EraseChar(h,p,15);
           GetValue(value,'0.01','1E5',real);
           Insert("IS",string,0);
           Insert(value,string,2);
           SetFRA(string);
           SetFRA('AUO          ');
        ELSIF value[0]="1" THEN
           RemoveSettingIfExisting("IS");
```

```
            p.v:=0.7;
            WriteString(h,p,"INTEGRATION TIME:");
            p.v:=0.65;
            EraseChar(h,p,15);
            WriteString(h,p,"VALUE IN CYCLES");
            p.v:=0.6;
            EraseChar(h,p,15);
            p.v:=0.55;
            EraseChar(h,p,15);
            GetValue(value,'1.0','1E6',real);
            Insert("IC",string,0);
            Insert(value,string,2);
            SetFRA(string);
            SetFRA('AU0          ');
         ELSIF value[0]="2" THEN    (* auto integration menu *)
            DrawRectangle(h,r);
            p.v:=0.7;
            EraseChar(h,p,16);
            p.v:=0.65;
            EraseChar(h,p,16);
            p.v:=0.6;
            EraseChar(h,p,16);
            p.v:=0.55;
            EraseChar(h,p,16);
            p.h:=1.05;
            p.v:=0.7;
            WriteString(h,p,"AUTO INTEGRATION");
            p.v:=0.66;
            WriteString(h,p,"Ch1 LONG  =1");
            p.v:=0.62;
            WriteString(h,p,"Ch2 LONG  =2");
            p.v:=0.58;
            WriteString(h,p,"Ch1 SHORT =3");
            p.v:=0.54;
            WriteString(h,p,"Ch2 SHORT =4");
            p.v:=0.50;
            WriteString(h,p,"OFF       =0");
            GetValue(value,'0','4',cardinal);
            IF value[0]="0" THEN
               SetFRA("AU0          ");
            ELSIF value[0]="1" THEN
               SetFRA("AU1          ");
            ELSIF value[0]="2" THEN
               SetFRA("AU2          ");
            ELSIF value[0]="3" THEN
               SetFRA("AU3          ");
            ELSIF value[0]="4" THEN
               SetFRA("AU4          ");
            END;
         END;
 END IntegrationSetting;


BEGIN
 menufinished:=FALSE;
 FillRectangle(h,rmax);
 REPEAT
   ShowSettings(bredd,hojd);
   SetTextColor(h,intensewhite);
   p.h:=bredd*31.0;
   p.v:=hojd*23.0*16.0/24.0;
   WriteString(h,p,"MANUAL INITIALISATION");
   SetTextColor(h,white);
   SetLineColor(h,red);
   p.h:=0.0;
   p.v:=14.0*hojd;
   WriteString(h,p,"GENERATOR:");
   p.v:=6.0*hojd;
   WriteString(h,p,"ANALYSER:");
   p.v:=11.0*hojd;
   WriteString(h,p,"SWEEP:");
```

```
 p.v:=5.0*hojd;
 p.h:=18.0*bredd;
 WriteString(h,p,"MEASUREMENT DELAY:");
 p.v:=6.0*hojd;
 WriteString(h,p,"INTEGRATION TIME :");
 p.v:=hojd;
 WriteString(h,p,"BACK TO MAIN MENU");
 p.v:=2.0*hojd;
 WriteString(h,p,"HELP");
 p.v:=9.0*hojd;
 WriteString(h,p,"LOG INCREMENT:");
 p.v:=10.0*hojd;
 WriteString(h,p,"MAX FREQUENCY:");
 p.v:=11.0*hojd;
 WriteString(h,p,"MIN FREQUENCY:");
 p.v:=12.0*hojd;
 WriteString(h,p,"WAVEFORM     :");
 p.v:=13.0*hojd;
 WriteString(h,p,"BIAS         :");
 p.v:=14.0*hojd;
 WriteString(h,p,"AMPLITUDE    :");
 r.loleft.h:=11.0*bredd;
 r.upright.h:=16.0*bredd;

FOR i:=2 TO 15 DO
 IF (i<>9) AND (i<>8) AND (i<>5) AND (i<>4) THEN
   r.loleft.v:=FLOAT(i-1)*hojd;
   r.upright.v:=FLOAT(i)*hojd-0.01;
   DrawRectangle(h,r);
 END;
END;
ShowCursor;
i:=WaitMouseRectangle(h);
r.loleft.v:=FLOAT(i-1)*hojd;
r.upright.v:=FLOAT(i)*hojd-0.01;
SetFillColor(h,red);
HideCursor;
FillRectangle(h,r);
SetFillColor(h,black);
FOR j:=0 TO 11 DO
   value[j]:=CHR(0);
   string[j]:=CHR(0);
END;

r.loleft.h:=1.025;    (* koordinater for inforuta *)
r.loleft.v:=0.5;
r.upright.h:=1.5;
r.upright.v:=0.75;
FOR j:=0 TO 11 DO
   string[j]:=CHR(0);
END;

CASE i OF
 2:    menufinished:=TRUE;
         (* back to main menu *)


 3:FillRectangle(h,rmax);
   p.h:=0.0;
   p.v:=0.95;
   WriteString(h,p,"HELP MANUAL INITIALISATION");
   p.v:=0.7;
   WriteString(h,p,"See 'SOLARTRON 1250 Operating Manual' (english),");
   p.v:=0.6;
   WriteString(h,p,"or 'PC-FRA, Manual' for help in swedish.");
   p.v:=0.5;
   WriteString(h,p,"The program needs some time between the measurements,");
   p.v:=0.4;
   WriteString(h,p,"so a minimum of MEASUREMENT DELAY is about 1 s.");
   p.h:=0.7;
   p.v:=0.1;
```

```
      WriteString(h,p,"PRESS MOUSE BUTTON TO RETURN");
      WaitForMouse(h,p,b);
|        (* help *)


   6:    DrawRectangle(h,r);
         p.h:=1.05;
         p.v:=0.7;
         WriteString(h,p,"MEASUREMENT DELAY");
         p.v:=0.65;
         WriteString(h,p,"  SECONDS=0");
         p.v:=0.6;
         WriteString(h,p,"  CYKLES =1");
         GetValue(value,'0','1',cardinal);
         IF value[0]="1" THEN
            RemoveSettingIfExisting("MS");
            p.v:=0.7;
            WriteString(h,p,"MEASUREMENT DELAY:");
            p.v:=0.65;
            EraseChar(h,p,15);
            WriteString(h,p,"VALUE IN CYCLES");
            p.v:=0.6;
            EraseChar(h,p,15);
            GetValue(value,'0.0','1E6',real);
            Insert("MC",string,0);
            Insert(value,string,2);
            SetFRA(string);
         ELSE
            RemoveSettingIfExisting("MC");
            p.v:=0.7;
            WriteString(h,p,"MEASUREMENT DELAY:");
            p.v:=0.65;
            EraseChar(h,p,15);
            WriteString(h,p,"VALUE IN SECONDS");
            p.v:=0.6;
            EraseChar(h,p,15);
            GetValue(value,'0.0','1E5',real);
            Insert("MS",string,0);
            Insert(value,string,2);
            SetFRA(string);
         END;

|          (* MEASUREMENT DELAY *)


   7:    IntegrationSetting(r);
|          (* integration *)
   8:|    (* nop *)
   9:|    (* nop *)

   10: (* LOG INCREMENT *)
         DrawRectangle(h,r);
         p.h:=1.05;
         p.v:=0.7;
         WriteString(h,p,"LOG INCREMENT");
         p.v:=0.65;
         WriteString(h,p,"STEPS/DECADE=0");
         p.v:=0.6;
         WriteString(h,p,"STEPS/SWEEP =1");
         GetValue(value,'0','1',cardinal);
         IF value[0]="1" THEN
            RemoveSettingIfExisting("GD");
            (* inmatningrutin for ny fordrojning *)
            p.v:=0.7;
            WriteString(h,p,"LOG INCREMENT:");
            p.v:=0.65;
            EraseChar(h,p,15);
            WriteString(h,p,"STEPS/SWEEP");
            p.v:=0.6;
            EraseChar(h,p,15);
            GetValue(value,'1.0','1E5',real);
```

```
            Insert("GS",string,0);
            Insert(value,string,2);
            SetFRA(string);
         ELSE
            RemoveSettingIfExisting("GS");
            p.v:=0.7;
            WriteString(h,p,"LOG INGREMENT:");
            p.v:=0.65;
            EraseChar(h,p,15);
            WriteString(h,p,"STEPS/DECADE");
            p.v:=0.6;
            EraseChar(h,p,15);
            GetValue(value,'0.33','1E5',real);
            Insert("GD",string,0);
            Insert(value,string,2);
            SetFRA(string);
         END;
|


   11:   DrawRectangle(h,r);
         p.h:=1.05;
         p.v:=0.6;
         WriteString(h,p,"MAX SWEEPING FREQUENCY ");
         GetValue(value,'1E-5','65535',real);
         Insert("MA",string,0);
         Insert(value,string,2);
         SetFRA(string);
|          (* maxfrekv *)


   12:   DrawRectangle(h,r);
         p.h:=1.05;
         p.v:=0.6;
         WriteString(h,p,"MIN SWEEPING FREQUENCY ");
         GetValue(value,'1E-5','65535',real);
         Insert("MI",string,0);
         Insert(value,string,2);
         SetFRA(string);
|          (* minfrekv *)


   13:   DrawRectangle(h,r);
         p.h:=1.05;
         p.v:=0.7;
         WriteString(h,p,"SETTING OF WAVEFORM");
         p.v:=0.65;
         WriteString(h,p,"  SINUS=   0");
         p.v:=0.6;
         WriteString(h,p,"  SQUARE=  1");
         GetValue(value,'0','1',cardinal);
         Insert("WF",string,0);
         Insert(value,string,2);
         SetFRA(string);
|          (* vagform *)


   14:   DrawRectangle(h,r);
         p.h:=1.05;
         p.v:=0.6;
         WriteString(h,p,"SETTING OF BIAS");
         GetValue(value,'-10.23','10.23',real);
         Insert("BI",string,0);
         Insert(value,string,2);
         SetFRA(string);
|          (* bias *)
   15:   AmplMenu(bredd,hojd);
  END;
  FillRectangle(h,rmax);
 UNTIL menufinished;
```

```
END ManualInitMenu;


PROCEDURE ChainMenu(bredd,hojd:REAL);
(* meny och rutiner for att satta enskilda parametrar i frekvensanalysatorn *)

VAR p:point;
    r:rectangle;
    i,j:CARDINAL;
    okay,menufinished:BOOLEAN;
    parameter:StringType;
    string:StringType;
    fil,copy_to:File;
    b:buttonset;
    filename:FileNameType;
    ch:CHAR;
BEGIN
 menufinished:=FALSE;
 FillRectangle(h,rmax);
 REPEAT
   SetTextColor(h,intensewhite);
   p.h:=bredd*31.0;
   p.v:=hojd*23.0*16.0/24.0;
   WriteString(h,p,"CHAIN");
   SetTextColor(h,white);
   SetLineColor(h,red);
   p.h:=18.0*bredd;
   p.v:=hojd;
   WriteString(h,p,"BACK TO MAIN MENU");
   p.v:=2.0*hojd;
   WriteString(h,p,"HELP");
   p.v:=10.0*hojd;
   WriteString(h,p,"SEE  CHAIN");
   p.v:=11.0*hojd;
   WriteString(h,p,"LOAD CHAIN");
   p.v:=12.0*hojd;
   WriteString(h,p,"SAVE CHAIN");
   p.v:=13.0*hojd;
   WriteString(h,p,"MAKE CHAIN");
   p.v:=14.0*hojd;
   WriteString(h,p,"ON/OFF");
   p.h:=27.0*bredd;
   IF chain_on THEN
      WriteString(h,p,'IS ON');
   ELSE
      WriteString(h,p,'IS OFF');
   END;
   r.loleft.h:=11.0*bredd;
   r.upright.h:=16.0*bredd;

  FOR i:=2 TO 15 DO
   IF (i>10) OR (i<4) THEN
     r.loleft.v:=FLOAT(i-1)*hojd;
     r.upright.v:=FLOAT(i)*hojd-0.01;
     DrawRectangle(h,r);
   END;
  END;
  ShowCursor;
  i:=WaitMouseRectangle(h);
  r.loleft.v:=FLOAT(i-1)*hojd;
  r.upright.v:=FLOAT(i)*hojd-0.01;
  SetFillColor(h,red);
  HideCursor;
  FillRectangle(h,r);
  SetFillColor(h,black);
  FOR j:=0 TO 11 DO
     string[j]:=CHR(0);
  END;

  CASE i OF
   2:    menufinished:=TRUE;
```

```
|         (* back to main menu *)
 3:FillRectangle(h,rmax);
   p.h:=0.0;
   p.v:=0.95;
   WriteString(h,p,"HELP CHAIN");
   p.v:=0.7;
   WriteString(h,p,"ON/OFF toggles CHAIN mode.");
   p.v:=0.6;
   WriteString(h,p,"MAKE CHAIN connects existing parameter files. ");
   p.v:=0.5;
   WriteString(h,p,"LOAD CHAIN copies an existing .CHN file into TEMP.CHN");
   p.v:=0.4;
   WriteString(h,p,"SAVE CHAIN renames TEMP.CHN another filename.CHN.");
   p.v:=0.3;
   WriteString(h,p,"SEE CHAIN types TEMP.CHN on the screen.");
   p.h:=0.7;
   p.v:=0.1;
   WriteString(h,p,"PRESS MOUSE BUTTON TO RETURN");
   WaitForMouse(h,p,b);
|        (* help *)
 6:|
 7:|
 8:|    (* nop *)
 9:|    (* nop *)
 10:|
 11:    DosCommand('cls','',okay);
        Lookup(fil,'TEMP.CHN',FALSE);
        REPEAT
          ReadChar(fil,ch);
          IF ch=EOL THEN
            Write(12C);
            Write(CHR(13));
          ELSE
            Write(ch);
          END;
        UNTIL (ch=EOF) OR (ch=CHR(0));
        Close(fil);
        WaitForMouse(h,p,b);
|        (* SEE *)
 12:
      p.h:=0.0;
      p.v:=0.55;
      DosCommand('cls','',okay);
      DosCommand('dir/w','*.CHN',okay);
      WriteString(h,p,'Which existing file.CHN to load?');
      GetFileName(filename,CHN);
        Lookup(fil,filename,FALSE);

                                (* 'notdone' is returned to fil.   *)
        Lookup(copy_to,'TEMP.CHN',TRUE);
        REPEAT
          ReadChar(fil,ch);
          WriteChar(copy_to,ch);
        UNTIL (ch=EOF) OR (ch=CHR(0));
        Close(fil);
        Close(copy_to);
|        (* LOAD *)
 13:
     DosCommand('cls','',okay);
     DosCommand('dir/w','*.chn',okay);
     GetFileName(filename,CHN);
     Lookup(fil,'TEMP.CHN',TRUE);
     Rename(fil,filename);
     Close(fil);
|        (* SAVE *)
 14:
     DosCommand('del','temp.chn',okay);
     Lookup(fil,'temp.chn',TRUE);
     REPEAT
       DosCommand('cls','',okay);
       DosCommand('dir/w','*.fra',okay);
```

```
        GetFileName(filename,FRA);
        FOR j:=1 TO 12 DO
          WriteChar(fil,filename[j]);
        END;
        WriteChar(fil,EOL);
        p.h:=0.1;
        p.v:=0.0;
        SetTextColor(h,black);
        WriteString(h,p,"  One more link? (Y/N)");
        SetTextColor(h,white);
        ReadString(h,p,string);
      UNTIL (string[0]='N') OR (string[0]='n');
      Close(fil);

|         (* MAKE *)
   15:  chain_on:=NOT chain_on;
  END;
  FillRectangle(h,rmax);
 UNTIL menufinished;
END ChainMenu;



PROCEDURE Send(ch: CHAR);
(* sander ett tecken till frekvensanalysatorn *)
BEGIN
  TransmitBuf^.text[0] := ch;
  TransmitBuf^.nx:=1;
  TransmitBuffer(TransmitBuf);
END Send;


PROCEDURE SendString(string:StringType;n:CARDINAL);
(*
  sander teckenstrang till frekvensanalysatorn med CR efter.
*)
VAR i:CARDINAL;
BEGIN
  FOR i:=0 TO n-1 DO
    TransmitBuf^.text[i] := string[i];
  END;
  TransmitBuf^.nx:=n;
  TransmitBuffer(TransmitBuf);
  Send(CHR(13));   (* command terminator *)
END SendString;



PROCEDURE InitFRA(filename:FileNameType);
(* Laser en fil fran disk, sander kommandona till frekvensanalysatorn *)
(* Lagger alla kommandona i settings-strangen, vilken anvands globalt i  *)
(* programmet for att veta FRAns status *)

VAR ch:CHAR;        (* tillfallig variabel *)
    f:File;
    i:CARDINAL;
    string:StringType;  (* samlar upp ett helt FRA kommando *)
    (* aven globala variabeln settings anvands *)
BEGIN

  Lookup(f,filename,FALSE);
(*
   I filen Default.fra finns alla frekvensanalysatorns defaultparametrar.
   Lookup assignar f till filename som en arbetsfil.
*)

  FOR i:=0 TO 200 DO
      settings[i]:=" ";
  END;
  Reset(f);
```

```
  i:=0;
  settingcounter:=0;

  REPEAT
    ReadChar(f,ch);

    IF ch=";" THEN
       INC(settingcounter);
       settings[settingcounter]:=ch;
       SendString(string,i);
       Send(CHR(13));  (* ; is command terminator *)
       FOR i:=0 TO 11 DO
         string[i]:=" ";
       END;
       i:=0;
    ELSIF
       (ORD(ch)=15B)  OR (ORD(ch)=12B) OR (ORD(ch)=32B) OR
       (ORD(ch)=36B) OR ( ch=" ") THEN;     (* NOP *)
(*
   Skippa alla kontrolltecken.
*)
    ELSE
       string[i]:=ch;          (* samla till helt FRA kommando *)
       INC(i);
       INC(settingcounter);
       settings[settingcounter]:=ch;

    END;
  UNTIL (ORD(ch)=32B) OR (ORD(ch)=0B);   (* until EOF or outside file *)
  Close(f);
END InitFRA;


PROCEDURE SetFRA(parameter:StringType);
VAR command:ARRAY [0..1] OF CHAR;
    i,j,position,newparameterlength,oldparameterlength:CARDINAL;
BEGIN

  command[0]:=parameter[0];
  command[1]:=parameter[1];
  newparameterlength:=0;
  FOR i:=0 TO 11 DO
     IF ((parameter[i]<="Z") AND (parameter[i]>="A")) OR
        ((parameter[i]<="9") AND (parameter[i]>="0")) OR
         (parameter[i]=".")  OR  (parameter[i]="-") THEN
            INC(newparameterlength);
     END;
  END;
  SendString(parameter,newparameterlength);

(* parameter was sent to the fra. Now it is time for the program *)
(* records i.e. adjusting "settings".                            *)

  parameter[newparameterlength]:=";"; (* command terminator added *)
  INC(newparameterlength);
  oldparameterlength:=0;
  position:=Pos(command,settings);
  IF position<=HIGH(settings) THEN
(*
   1) Om typen av kommando tidigare ar satt skrivs det over av det nya
      kommandot om det har lika manga tecken.
   2) Har de olika manga tecken tas det gamla kommandot bort och det
      nya laggs in sist.
   3) Ar kommandotypen inte tidigare satt laggs den sist.
*)
     i:=position;
     REPEAT
       INC(oldparameterlength);
       INC(i);
     UNTIL settings[i-1]=";";
     IF oldparameterlength = newparameterlength THEN
```

```
        FOR i:=0 TO newparameterlength-1 DO
            settings[i+position]:=parameter[i];
        END;
      ELSE
        FOR i:=0 TO settingcounter-position-oldparameterlength DO
            settings[position+i]:=settings[position+i+oldparameterlength];
        END;
        settingcounter:=settingcounter-oldparameterlength;
        FOR i:=1 TO newparameterlength DO
            settings[settingcounter]:=parameter[i-1];
            INC(settingcounter);
        END;
        FOR i:=settingcounter TO 200 DO
            settings[i]:=" ";
        END;
      END;
    ELSE
      FOR i:=1 TO newparameterlength DO
          settings[settingcounter]:=parameter[i-1];
          INC(settingcounter);
      END;
      FOR i:=settingcounter TO 200 DO
          settings[i]:=" ";
      END;
    END;

END SetFRA;


PROCEDURE WriteSettings(filename:FileNameType);
(* skriver FRA-parametrarna pa fil for senare bruk *)
VAR f       :File;
    ch      :CHAR;
    i       :CARDINAL;
BEGIN
  Lookup(f,filename,TRUE);
  Reset(f);
  i:=0;
  REPEAT
    INC(i);
    ch:=settings[i];
    IF ch<>" " THEN
       WriteChar(f,ch);
    END;
  UNTIL (ch=" ") OR (i=200);
  Close(f);

END WriteSettings;


 PROCEDURE ExtractValue(VAR s:StringType;i:CARDINAL);
(* workroutine for search in 'settings' .*)
 VAR j:CARDINAL;
 BEGIN
   FOR j:=0 TO 11 DO
     s[j]:=CHR(0);
   END;
   j:=0;
   i:=i+2;
   REPEAT
     s[j]:=settings[i];
     INC(i);
     INC(j);
   UNTIL settings[i]=';';
 END ExtractValue;


PROCEDURE ShowSettings(bredd,hojd:REAL);
(* Writes actual settings om the screen *)
VAR i:CARDINAL;
    s:StringType;
```

```
    p:point;


BEGIN
  s[0]:='0';

  i:=Pos('AP',settings);
  IF i<settingcounter THEN
    ExtractValue(s,i);
    IF s[0]="1" THEN   (* compression on *)
      p.v:=14.0*hojd;
      p.h:=37.0*bredd;
      WriteString(h,p,'COMPRESSION ON');
    END;
  END;
  IF (i>settingcounter) OR (s[0]='0') THEN
    i:=Pos('AM',settings);
    IF i<settingcounter THEN
       ExtractValue(s,i);
       p.v:=14.0*hojd;
       p.h:=37.0*bredd;
       WriteString(h,p,s);
       p.h:=42.0*bredd;
       WriteString(h,p,'V');
    END;
  END;

  i:=Pos('BI',settings);
  IF i<settingcounter THEN
     ExtractValue(s,i);
     p.v:=13.0*hojd;
     p.h:=37.0*bredd;
     WriteString(h,p,s);
     p.h:=42.0*bredd;
     WriteString(h,p,'V');
  END;

  i:=Pos('MI',settings);
  IF i<settingcounter THEN
     ExtractValue(s,i);
     p.v:=11.0*hojd;
     p.h:=37.0*bredd;
     WriteString(h,p,s);
     p.h:=42.0*bredd;
     WriteString(h,p,'Hz');
  END;

  i:=Pos('MA',settings);
  IF i<settingcounter THEN
     ExtractValue(s,i);
     p.v:=10.0*hojd;
     p.h:=37.0*bredd;
     WriteString(h,p,s);
     p.h:=42.0*bredd;
     WriteString(h,p,'Hz');
  END;

  i:=Pos('MS',settings);
  IF i<settingcounter THEN
     ExtractValue(s,i);
     p.v:=5.0*hojd;
     p.h:=37.0*bredd;
     WriteString(h,p,s);
     p.h:=42.0*bredd;
     WriteString(h,p,'s');
  END;

  i:=Pos('GD',settings);
  IF i<settingcounter THEN
     ExtractValue(s,i);
     p.v:=9.0*hojd;
```

```
   p.h:=37.0*bredd;
   WriteString(h,p,s);
   p.h:=42.0*bredd;
   WriteString(h,p,'steps/decade');
END;

i:=Pos('GS',settings);
IF i<settingcounter THEN
   ExtractValue(s,i);
   p.v:=9.0*hojd;
   p.h:=37.0*bredd;
   WriteString(h,p,s);
   p.h:=42.0*bredd;
   WriteString(h,p,'steps/sweep');
END;


i:=Pos('MC',settings);
IF i<settingcounter THEN
   ExtractValue(s,i);
   p.v:=5.0*hojd;
   p.h:=37.0*bredd;
   WriteString(h,p,s);
   p.h:=42.0*bredd;
   WriteString(h,p,'cycles');
END;


i:=Pos('WF',settings);
IF i<settingcounter THEN
   ExtractValue(s,i);
   p.v:=12.0*hojd;
   p.h:=37.0*bredd;
   IF s[0]='0' THEN
     WriteString(h,p,'sinus');
   ELSE
     WriteString(h,p,'square');
   END;
END;

i:=Pos('AU',settings);
IF i<settingcounter THEN
   ExtractValue(s,i);
   p.v:=6.0*hojd;
   p.h:=37.0*bredd;
   IF (s[0]<>'0') THEN
    WriteString(h,p,'auto');
   END;
   p.h:=42.0*bredd;
   IF (s[0]='1') THEN
     WriteString(h,p,'Ch1 long');
   ELSIF (s[0]='2') THEN
     WriteString(h,p,'Ch2 long');
   ELSIF (s[0]='3') THEN
     WriteString(h,p,'Ch1 short');
   ELSIF (s[0]='4') THEN
     WriteString(h,p,'Ch2 short');
   ELSIF s[0]='0' THEN
      i:=Pos('IS',settings);
      IF i<settingcounter THEN
        ExtractValue(s,i);
        p.v:=6.0*hojd;
        p.h:=37.0*bredd;
        WriteString(h,p,s);
        p.h:=42.0*bredd;
        WriteString(h,p,'s');
     END;

     i:=Pos('IC',settings);
     IF i<settingcounter THEN
        ExtractValue(s,i);
```

```
          p.v:=6.0*hojd;
          p.h:=37.0*bredd;
          WriteString(h,p,s);
          p.h:=42.0*bredd;
          WriteString(h,p,'cycles');
        END;
      END;
  END;

END ShowSettings;




PROCEDURE AmplValues(bredd,hojd:REAL);
(* writes present settings on screen in AmplMenu *)
VAR i:CARDINAL;
    s:StringType;
    p:point;

BEGIN
  i:=Pos('AP',settings);
  IF i<settingcounter THEN
    ExtractValue(s,i);
    p.v:=12.0*hojd;
    p.h:=47.0*bredd;
    IF s[0]="1" THEN   (* compression on *)
      WriteString(h,p,'ON');
    ELSE
      WriteString(h,p,'OFF');
    END;
  END;

    i:=Pos('AM',settings);
    IF i<settingcounter THEN
       ExtractValue(s,i);
       p.v:=14.0*hojd;
       p.h:=37.0*bredd;
       WriteString(h,p,s);
       p.h:=42.0*bredd;
       WriteString(h,p,'V');
    END;

    i:=Pos('AV',settings);
    IF i<settingcounter THEN
       ExtractValue(s,i);
       p.v:=10.0*hojd;
       p.h:=37.0*bredd;
       WriteString(h,p,s);
       p.h:=42.0*bredd;
       WriteString(h,p,'V');
    END;

    i:=Pos('AE',settings);
    IF i<settingcounter THEN
       ExtractValue(s,i);
       p.v:=9.0*hojd;
       p.h:=37.0*bredd;
       WriteString(h,p,s);
       p.h:=42.0*bredd;
       WriteString(h,p,'%');
    END;


  i:=Pos('AS',settings);
  IF i<settingcounter THEN
    ExtractValue(s,i);
    IF s[0]="1" THEN
      p.v:=11.0*hojd;
```

```
      p.h:=37.0*bredd;
      WriteString(h,p,'Channel 1');
    ELSE
      p.v:=11.0*hojd;
      p.h:=37.0*bredd;
      WriteString(h,p,'Channel 2');
    END;
  END;

END AmplValues;


PROCEDURE StartUp;
(* startup routine called only once by the main program *)
(* Leaves over the command to HMeny after initialising  *)

VAR temp:FileNameType;
    i   :CARDINAL;

BEGIN
  finished:=FALSE;
  rmax.loleft.h:=0.0;    (* global whole screen rectangle *)
  rmax.loleft.v:=0.0;
  rmax.upright.h:=1.5;
  rmax.upright.v:=1.0;

  (* window initialisation *)
  VirtualScreen(h);
  SetWindow(h,rmax);
  SetViewPort(h,rmax);
  chain_on:=FALSE;
  p.h:=0.12;
  p.v:=0.5;
  WriteString(h,p,'CONTROL PROGRAM FOR SOLARTRON 1250 FREQENCY RESPONSE ANALYSE
R');
  p.h:=0.5;
  p.v:=0.4;
  WriteString(h,p,'initialising the analyser');
  string:="default.fre";
  FOR i:=0 TO 11 DO
    temp[i+1]:= string[i];
  END;
  InitFRA(temp);
  HMeny(finished);
END StartUp;


BEGIN (* main program *)
  SetPriority(20);
  StartUp;
  LOOP
    IF finished THEN (* satts i body/hmeny *)
       EXIT;
    END;
  END;
END PCFRA.
```